НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Інститут прикладного системного аналізу

Кафедра математичних методів системного аналізу

До захисту допущено:
В.о. завідувач кафедри
_____ Оксана ТИМОЩУК
«___»_____20__ р.

Дипломна робота
на здобуття ступеня бакалавра
за освітньо-професійною програмою «Системи та методи штучного інтелекту»
спеціальності 122 «Комп'ютерні науки та інформаційні технології»
на тему: «Моделі та методи інтелектуального аналізу даних COVID – 19»

Виконали:
студент IV курсу, групи КА-66
Сапєльніков Олександр Сергійович _____

студентка IV курсу, групи КА-66
Лупаненко Софія Олександрівна _____

_____

Керівник:
доцент, д.т.н. Недашківська Надія Іванівна _____

Консультант з нормоконтролю:
доцент, к.т.н. Коваленко Анатолій Єпіфанович _____

Рецензент:
доцент, к.т.н. кафедри СП ІПСА,
Безносик Олександр Юрійович

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
Студент _____

Київ 2020

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Інститут прикладного системного аналізу
Кафедра математичних методів системного аналізу

Рівень вищої освіти – перший (бакалаврський)
Спеціальність – 122 «Комп'ютерні науки та інформаційні технології»
Освітньо-професійна програма «Системи та методи штучного інтелекту»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ Оксана ТИМОЩУК
«25» травня 2020 р.

ЗАВДАННЯ
на дипломну роботу студентам
Сапєльнікову Олександру Сергійовичу.
Лупаненко Софії Олександрівні

1. Тема роботи «Моделі та методи інтелектуального аналізу даних COVID-19», керівник роботи доцент кафедри ММСА, Недашківська Надія Іванівна, затверджені наказом по університету від «25» травня 2020 р. № 1143-с

2. Термін подання студентами роботи 8.06.2020.

3. Вихідні дані до роботи : вибірка рентген зображень здорової та хворої на коронавірус людини, вибірка випадків захворювань на коронавірус 22.01.2020-10.05.20 .

4. Зміст роботи: Робота складається з трьох частин. ЧАСТИНА 1 - спільна частина. ЧАСТИНА 2 - аналітичний та історичний огляд рентген зображень, теоретичний огляд побудови згортковий нейронної мережи, прогнозування наявності захворювання, функціонально-вартісний аналіз

програмного продукту. ЧАСТИНА 3 - аналітичний огляд коронавірусу, теоретичні відомості та основі регресійного аналізу, розгляд методу опорних векторів, застосування нейронних мереж, ансамблі моделей та прогнозування поширення епідемії на основі вхідних даних, функціонально-вартісний аналіз програмного продукту.

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо) рентген зображення, архітектури згортових нейронних мереж, принцип роботи програмного продукту, результати прогнозування моделей, часові ряди, багатошарова нейрона мережа, принцип роботи програмного продукту, результати прогнозування моделей.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		зав дання видав	зав дання прийняв
Економічний	Шевчук О.А.		

7. Дата видачі завдання 27 березня 2020

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1.	Вивчення літератури за темою роботи.	13.04.2020	
2.	Підготовка першого розділу.	21.04.2020	
3.	Підготовка другого розділу.	1.05.2020	
4.	Розробка програмного продукту.	19.05.2020	
5.	Підготовка третього розділу	25.05.2020	
6.	Підготовка економічної частини	28.05.2020	
7.	Оформлення розділів відповідно до		

	нормоконтролю.		
8.	Підготовка презентації доповіді.	30.05.2020	
9.	Оформлення дипломної роботи.		

Студент Олександр САПЄЛЬНІКОВ

Керівник Надія НЕДАШКІВСЬКА

РЕФЕРАТ

Дипломна робота містить: 249 с., 23 табл., 129 рис., 2 дод. та 36 джерел.

РЕГРЕСІЯ, НЕЙРОННА МЕРЕЖА, ІНТЕЛЕКТУАЛЬНИЙ АНАЛІЗ ДАНИХ, ЗГОРТКА, ПРОГНОЗУВАННЯ НАЯВНОСТИ ХВОРОБИ, РЕНТГЕН ЗОБРАЖЕННЯ, КЛАСИФІКАЦІЯ, ПРОГНОЗУВАННЯ ЧАСОВИХ РЯДІВ.

Об’єктом дослідження 2 частини є вибірка рентген зображень здорової людини та хворої на коронавірус.

Об’єктом дослідження 3 частини є вибірка захворюваності та смертності на коронавірус.

Предметом дослідження 2 частини є методи інтелектуального аналізу даних на основі згортокої нейронної мережи та класифікації зображень.

Предметом дослідження 3 частини є методи інтелектуального аналізу даних на основі регресії та аналізу часових рядів.

Програмною мовою була обрана Python.

В даній роботі проведено дослідження можливості класифікації рентген зображень та проаналізовано часові ряди. Для побудови моделей були використані згортові нейронні мережи в випадку 2 частини та регресійні методи на основі простих нейронних мереж в випадку 3 частини. Прогнозування було виконано на основі рентген зображень здорових та хворих людей в випадку 2 частини та основі часових рядів захворювання та смертності в випадку 3 частини .

При виконанні роботи було встановлено моделі та методи, що дають найкращі результати які достатньо близькі до реальних. Напрямок розвитку роботи є в розширенні функціоналу та зменшенні похибки прогнозування. Планується додати можливість класифікації типу коронавірусу та встановити залежність між смертністю та захворюваністю у різних країнах світу.

ABSTRACT

Thesis: 249p, 23 tabl., 129 fig., 2 add. and 36 references.

REGRESSION, NEURAL NETWORK, INTELLECTUAL DATA ANALYSIS, CONVOLUTION, FORECASTING THE PREDICT OF THE DISEASE, X-RAY IMAGE, CLASSIFICATION, TIME SERIES FORECASTING.

The object of the research of Part 2 is a sample of X-ray images of a healthy person and a patient with coronavirus.

The object of the research of Part 3 is the sampling of morbidity and mortality from coronavirus.

The subject of the research of Part 2 are methods of data mining based on convoluted neural network and image classification.

The subject of the research of Part 3 are methods of data mining based on regression and time series analysis.

Programming language Python.

This work is a study of the possibility of classification of X-ray images and time series analysys. Collapsible neural networks in the case of part 2 and regression methods based on simple neural networks in the case of part 3 were used to build the models. The prediction was performed on the basis of X-ray images of healthy and sick people in the case of Part 2 and on the basis of time series of morbidity and mortality in the case of Part 3.

When performing the work, models and methods were established that give the best results that are close enough to the real ones. The direction of development of work is in expansion of functionality and reduction of forecasting error. It is planned to add the possibility of classifying the type of coronavirus and to establish a relationship between mortality and morbidity in different countries.

ЗМІСТ

ЧАСТИНА 1

ЧАСТИНА 2

ЧАСТИНА 3

ВСТУП

На сьогодні, в кожній країні головною ціллю є протистояння пандемії. Вона перевернула життя кожного з нас. Тому головною задачею цієї роботи став аналіз наслідків COVID-19.

Головне джерело аналізу – це дані. Вони дуже важливі, адже без них зрозуміти хворобу неможливо. Лише завдяки правильних даних ми зможемо побачити, як вона поширюється та який вплив має на людей у всьому світі. Незалежно від обставин чи часу, аналіз та прогнозування є важливою допомогою для більшої ефективності планування майбутнього.

РОЗДІЛ 1 АНАЛІТИЧНИЙ РОЗГЛЯД COVID-19

1.1 Історія виникнення

Коронавіруси було виявлено з носової порожнини в 1960 році. Надалі ці зразки отримали назву коронавірус людини 229E і OC43.

Перший випадок захворювання людини вірусом COVID-19 зафіксували в Китаї, а саме в місті Ухань. Місто славиться своїм оптовим ринком морепродуктів та тварин – Хуанан. Саме перша хвиля уражених, мали з ним зв’язок. В минулому спалахи пандемій, наприклад SARS, були пов’язані з ринками тварин. Провідний імунолог Професор Стенлі Перлман, котрий є фахівцем з попередніх спалахів коронавірусу, що виникали від тварин, вважає, що ідея про зв'язок із ринком Ухань може бути випадковою, але це мало імовірно, оскільки генетичний матеріал вірусу був знайдений у ринковому середовищі.

На сьогодні, ніхто не може дати сто відсоткову інформацію, щодо того, де саме виник вірус, але можна припустити той факт, що людина отримала зараження від тварини. Прикладом став випадок в Нью-Йоркському зоопарку, де вірусом заразився тигр. Це говорить про те, що вірус може переміщатися між видами. Тоді, коли дослідники зможуть звузити перелік видів тварин. Ми зрозуміємо, які з видів є вразливими.

Вчені стверджують, що існує велика ймовірність походження вірусу від кажанів, що вперше пройшов через тварину-посередника до людини. Саме таке походження було у SARS 2002 року. Коли підковонісний кажан передав вірус людині через цивету (рис. 1.1).

Рисунок 1.1 – Цивета і кажан – посередники.

Імунолог CSIRO – Доктор Мішель Бейкер, вивчає віруси кажанів. Вона говорить, що деякі дослідження походження COVID-19 відійшли від того, що було відомо з минулого. Наскільки точна ця історія походження ми не знаємо, але те, що якийсь зв'язок з ринком Ухань цілком існує.

Є сотні коронавірусів, більшість з яких циркулює серед таких тварин, як свині, верблюди, кажани та коти. Іноді ці віруси стрибають до людини – це називається побічною реакцією і можуть спричинити захворювання.

Чотири з семи відомих, якими хворіють люди, викликають лише легку та помірну хворобу. Три можуть спричинити більш серйозні, навіть летальні, захворювання. SARS-CoV з'явився в листопаді 2002 року і викликав важкий гострий респіраторний синдром (ГРВІ). Цей вірус зник у 2004 році. Близькосхідний респіраторний синдром MERS (MERS-CoV). Перенесений з водойми тварин у верблюдів, MERS був виявлений у вересні 2012 року і продовжує викликати спорадичні та локалізовані спалахи. Третій новий коронавірус, що з'явився у цьому столітті, називається SARS-CoV-2. Саме він викликає коронавірусну хворобу 2019 (рис. 1.2).

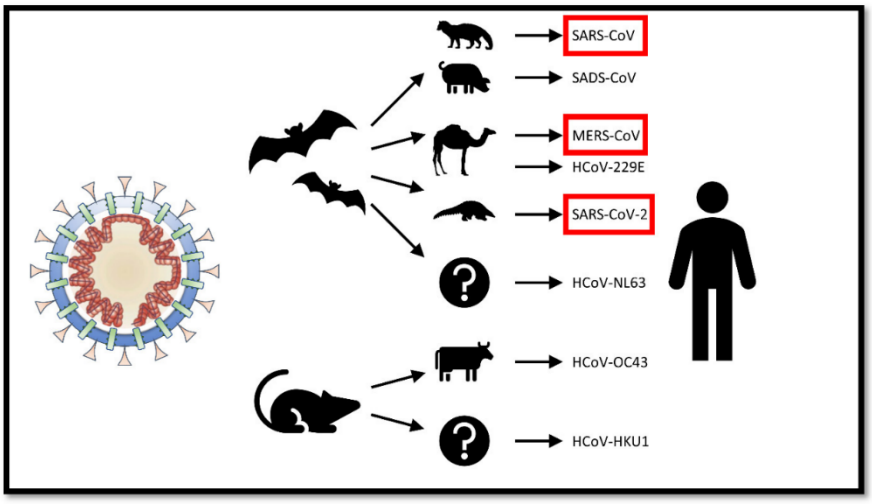


Рисунок 1.2 – Походження коронавірусів

На рисунках 1.3 – 1.5 зображено як розвивались COVID-19 та SARS. За даними ВООЗ, понад 8000 людей були заражені SARS протягом усього спалаху, а близько 800 померли. Це означає, що приблизно один з 10 хворих

був убитий хворобою. Кількість смертей та випадків захворювання, від сучасного коронавірусу, з кожним днем зростає експоненціально.

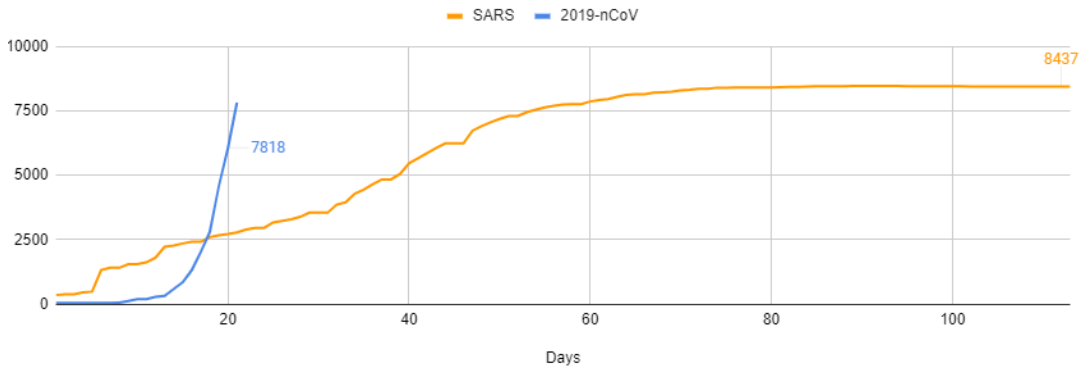


Рисунок 1.3 – Кількість заражень по днях

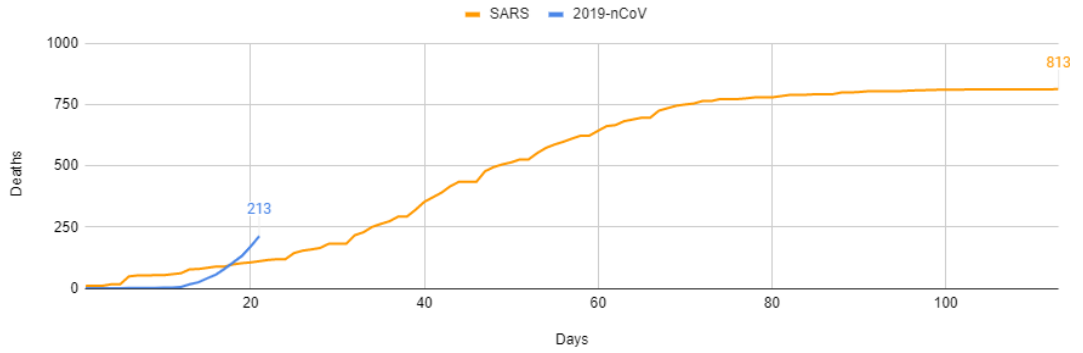


Рисунок 1.4 – Кількість смертей по днях

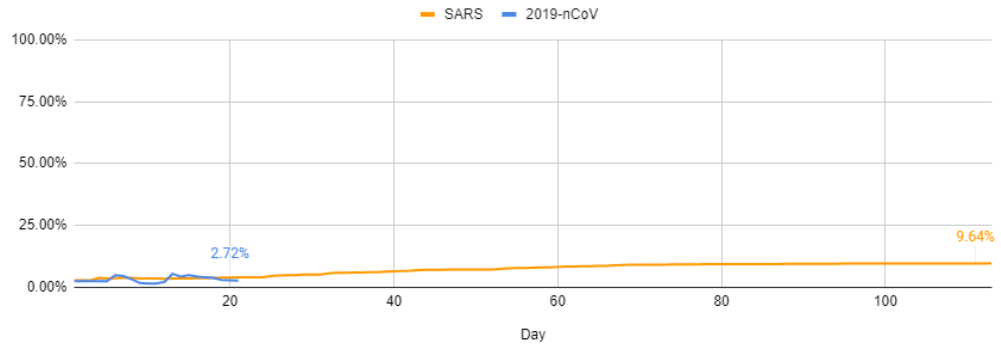


Рисунок 1.5 – Кількість смертей/інфекцій по днях

На березень місяць, пандемія перебуває на стадії розробки, після того як Covid-19 поширився з міста Хуань по провінції Хубей в Китай, а потім по всьому світу. Існують невизначеності щодо декількох аспектів історії походження Covid-19, які вчені намагаються розгадати, в тому числі, який

вид передав його людині. Адже знання того, як почалась пандемія – це ключ до її зупинки.

1.2 Визначення COVID-19

Коронавіруси – це віруси, які циркулюють серед тварин, а деякі з них також, як відомо, заражають людину. На сьогодні відомо також те, що кілька видів тварин виступають джерелами. У людей, зазвичай, вони викликають респіраторні інфекції, починаючи від звичайної застуди до більш важких захворювань. Останній виявлений коронавірус викликає – COVID-19. Абревіатуру обрала ВООЗ, котра означає «коронавірусна хвороба 2019».

Ситуація котру ми маємо сьогодні – це набагато більше, ніж криза здоров'я. Підкреслюючи кожну з країн, до яких вона торкається, вона має потенціал створити руйнівні соціальні, економічні та політичні кризи, які залишать глибокі шрами.

Тим часом поширюються різні божевільні теорії змови про те, що вірус якимось чином вирвався з китайської лабораторії. Однак це категорично неправда, адже вчені проаналізували геномну різноманітність вірусу SARS-CoV-2, який викликає COVID-19, шляхом скринінгу геномів понад 7500 вірусів, отриманих із зразків декількох сотень тисяч пацієнтів з підтвердженим зараженням інфекцією з усього світу. В результаті дослідникам вдалося виявити 198 мутацій, тобто він не менше ніж на 70 % схожий за генетичною послідовністю на вірус SARS-CoV, що викликає атипову пневмонію, має особливий патогенез, оскільки уражає як верхні, так і нижні дихальні шляхи, а також може спричиняти гастроентерит. Згодом встановили генетичну послідовність геному нового вірусу.

Доцент кафедри імунології та мікробіології зазначив, що, порівнюючи дані послідовності генома для відомих видів короновіруса, ми можемо точно визначити, що SARS-CoV-2 виник в результаті природних процесів.

Також дослідили генетичний шаблон білкових шипів на зовнішній стороні сучасного вірусу, котрий використовує їх для проникнення в людину чи тварину через зовнішні стінки клітин.

Існують дві особливості білкового шипа:

- «Захватний гачок» як рецептор, котрий зв’язує домен – RBD;
- «Молекулярний ніж», який зламує стінку клітини і проникає в неї.

З’ясували, що RBD – це частина білків SARS-CoV-2, яка еволюціонувала, щоб ефективно впливати на молекулярну особливість клітин людини, а рецептор ACE2, який бере участь в регуляції кров'яного тиску. Білок SARS-CoV-2 був настільки ефективний для зв'язування людських клітин, що вчені прийшли до висновку, що це результат природного відбору, а не продукт генної інженерії.

З досліду на електронному мікроскопі (рис. 1.6) ми бачимо вірус SARS-CoV-2, який викликає COVID-19, виділений у пацієнта. Частинки вірусу виходять з поверхні клітин, культивованих у лабораторії. Шипи на зовнішньому краю вірусних частинок дають свою назву – коронаподібні. Ці шипи фіксуються на клітинах людини, після чого зазнають структурних змін, що дозволяють вірусної мембрані зливатися з клітинною мембраною. Потім вірусні гени можуть потрапляти в клітину-господаря для копіювання, виробляючи більше вірусів.

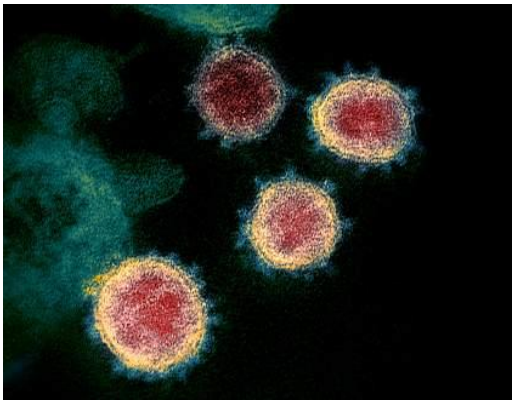


Рисунок 1.6 – Структура SARS-CoV-2

Станом на 10 травня 2020 р. за статистикою, кількість хворих перейшло за 4 мільйони, смертельних випадків досягло 280 тисяч, а одужали 1,4 мільйона людей.

1.3 Методи тестування COVID-19 у людини

Після першого різкого спалаху пандемії SARS COVID-19 в таких країнах Європи, як Іспанія та Італія, головним повідомленням від усіх експертів з охорони здоров'я було: "Тест, тест, тест". Обґрунтовуючи ці слова тим, що найкращій спосіб боротьбі з пандемією є згладжування кривої передачі вірусу. Тому експерти впевнені, що масове тестування дасть спромогу виявити позитивні результати, та відслідкувати їх контакти, сприяючі подальшої передачі.

На разі існує два розповсюджених видів тестування вірусу у людини.

- NAAT-тести;
- Серологічні тести.

NAAT-тести є найбільш використовувані, це рекомендовані ВОЗ тести на посилення нуклеїнової кислоти, які виявляють вірус SARS-CoV-2, що відповідає за захворювання COVID-19.

Зразок, як правило, збирають з верхніх дихальних шляхів за допомогою техніки носоглоткового мазка – в якій з горла за носом збирають пробу, що містить суміш слизової та слини. Зразки доставляють у спеціалізовану лабораторію та перевіряють на SARS-CoV-2, використовуючи аналіз ланцюгової реакції полімеразної зворотної транскрипції в режимі реального часу (rRT-PCR) – метод, що використовується для виявлення наявності "специфічного генетичного матеріалу від збудника". Зі слів керівника відділу алергії та інфекційних захворювань Вашингтонського університету, Анни Уолд, ці тести були виробленні в дуже авторитетних лабораторіях і техніка їх

створення не потребує жодних вдосконалень. Процес такого тестування може займати від 3 до 4 годин, а результати будуть доступні протягом декількох днів.

Для виявлення антитіл, наявних у сироватці крові, використовується серологічний тест. Антитіла – це білки, що виробляються білими кров'яними клітинами для боротьби з чужорідними тілами, такими як антигени. За даними університету Джона Хопкінса, «сироватка містить антитіла до конкретних компонентів патогенів, які називаються антигенами, визнані імунною системою як чужі та націлені на імунну відповідь». Автори тестів стверджують, що серологічні тести можуть допомогти виявити рівень захворювання в громаді, а отже, і рівень смертності від цієї хвороби.

Також, що є не мало важливим, дослідження дає змогу виявляти осіб, яка виробила антитіла до SARS-CoV-2, що дозволяє бути донором, для тих хто інфікований вірусом. Але ці тести потребують більших досліджень, так як на разі вони кажуть лише про можливу інфекцію або її опромінення в минулому, але не якщо ви зараз заразні. 2 квітня Управління з харчових продуктів та лікарських засобів США затвердило перший серологічний тест, виготовлений біотехнологічною компанією Cellex, що проводила тестування на антитіла до коронавірусу в крові, отриманої з пальця.

Як стверджує компанія, тест може виявити, чи є у людини COVID-19 приблизно за 15 хвилин.

На разі тестування проходить майже у всьому світі. Пріоритетність тестування залишається за людьми, які входять в групу ризику хвороби, зокрема це особи старше 60 років та люди з хронічними захворювання дихальних шляхів. Окрім людей, які в зоні ризику, тестуються усі працівники медичних закладів та інші, хто сприяє активній боротьбі з вірусом та знаходиться в осередку розповсюдження.

У таких країнах світу, як Іспанія, США, Італія, Росія, Франція ,де вірус розповсюдився до дуже великих масштабів, людина з легкими симптомами навряд чи зможе пройти тестування, тому що кількість якісних тестів є

обмеженою. ВОЗ рекомендує, у випадках виявлення інфікованого, одразу з’ясовувати, з ким ця осіб входила у контакт, та обов’язково робити тести цьому кругу людей.

Якщо говорити про місця, де можна пройти тестування, то по-перше це спеціалізовані лабораторії та медичні установи, такі як лікарні та клініки. Однак наприкінці березня, початку квітня, такі країни як Німеччина, Канада, Об'єднані Арабські Емірати та Південна Корея, відкрили центри тестування поза традиційними лікарняними установами. У Південній Кореї в багатьох містах почали проводити дорожні тести, які дали можливість скоротити час тестування приблизно на третину, а також знизивши ризик заразитися вірусом – оскільки людина знаходиться у своєму транспортному засобі. Вважається, що весь процес займає приблизно 10 хвилин.

Професор з Медичного коледжу університету Йонсея в Сеулі, Лі Хюкмін,
завила – «Це хороше рішення для швидкого відбору проб. У Кореї процес відбору проб дуже трудомісткий через проблеми біобезпеки для захисту медичного персоналу. Отже, достатньо випробувань не можна зробити навіть при ранньому розширенні лабораторних можливостей".

Кілька компаній у США стверджують, що створили домашні коронавірусні набори.

Однак, Американська агенція з контролю за продуктами харчування та лікарськими засобами наполегливо стверджує, що не " санкціонувала жоден тест, який можна придбати для тестування себе вдома на COVID-19".

У Бангладеш дослідники випустили тестовий набір, який, за їхніми словами, може виявити антитіла до SaRs-CoV-2 за 15 хвилин. Набори, затверджені урядом Бангладеші на виробництво, коштували всього 3 долари кожен.

Дослідники Сенегалу оголосили наприкінці березня новий домашній тест на коронавірус, який коштує приблизно 1 долар США – і може виявити активну інфекцію за допомогою мазка слини, або раніше не виявленого

випадку, використовуючи аналіз крові в домашніх умовах за допомогою пальця, котрий визначає антитіла до вірус.

Спеціаліст з інфекційних хвороб університетської мережі охорони здоров'я в Торонто Абду Шаркаві вважає, що набори для дому дозволять значно розширити тестування. Але він додав, що домашні тести, швидше за все, "менш чутливі, ніж лабораторні тести".

До країн які провели якісне масове тестування можна віднести Німеччину та Південну Корею, вони отримали широку оцінку суспільства, за їх зусилля до агресивного тестування, ці країни заявили що можуть проводити до 500 000 тестувань на день. Однак для багатьох країн, особливо тих, які мають середній чи низький рівень доходу, масштабне тестування є досить не можливим.

На рисунку 1.7 показана кількість тестів на тисячу осіб щодня, у протилежно різних за фінансовим становищем країнах світу. Важливо взяти до уваги що в різних країнах світу ця статистика рахується по різному, десь рахується кількість зроблених тестів, десь кількість унікальних протестованих людей.

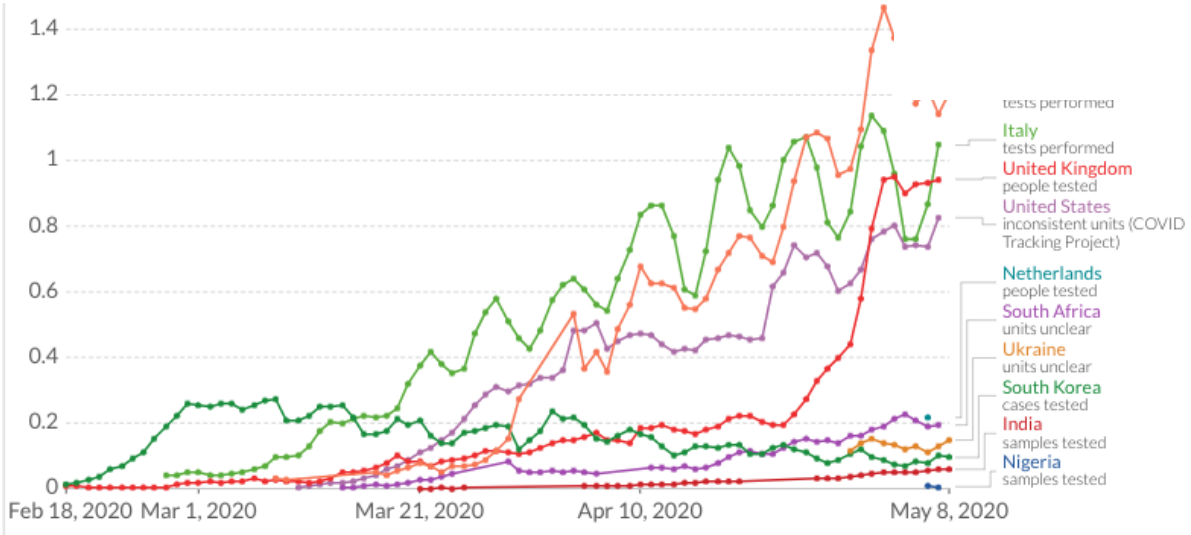


Рисунок 1.7 – тестування населення

На рисунку 1.8 можна побачити, які методи підрахунку, використовують розглянуті країни, та відносну кількість за 7 травня 2020 року.

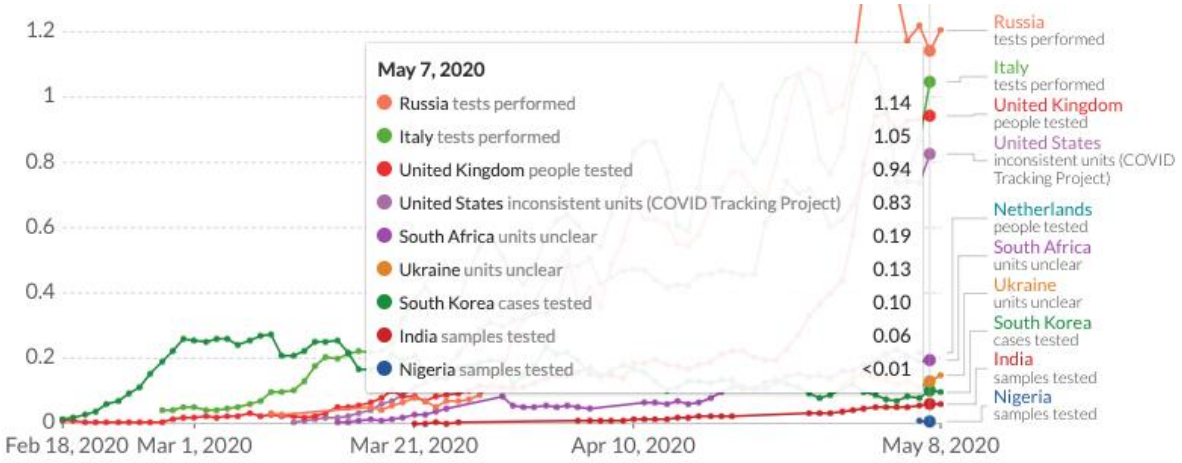


Рисунок 1.8 – методи підрахунку

1.4 Фактори, які впливають на протікання хвороби.

Більша кількість випадків зараження, відчувають легкі та помірні захворювання органів дихання, котрі не вимагають спеціального лікування. Суб'єкт може навіть не здогадуватись, що знаходиться в інкубаційному періоді або ж навіть хворий. Але все одно буде носієм хвороби і заражати інших. Вірус поширюється в основному через краплі слини або виділення з носа. Тобто, коли інфікована особа кашляє або чхає.

В зоні ризику знаходяться ті, хто мають:

- похилий вік;
- слабкий імунітет;
- діабет;
- хронічні респіраторні захворювання;
- рак
- і тд.

Згідно статистики США в таблиці 1.1 – 1.2, люди в зоні ризику мають найбільшу вірогідність ускладнення свого стану, що може призвести до смерті.

Основними хворобами були: діабет, хвороби легенів, рак, імунодефіцит, хвороби серця, гіпертонія, астма, захворювання нирок та захворювання ШКТ/печінки.

Таблиця 1.1 – Статистика смертності за віком в США на 14 квітня

Вік	**Кількість померлих**	**% смертей**	**З основними симптомами**	**Без основних симптомів**	**Симптоми невідомі**	**Частка загиблих невідомих**
0 – 17	3	0,04%	3	0	0	0%
18 – 44	309	4,5%	244	25	40	1,0%
45 – 64	1581	23,1%	1343	59	179	3,5%
65 – 74	1683	24,6%	1,272	26	385	6,0%
75+	3263	47,7%	2289	27	947	14,2%

Стать	**Смертей**	**%**	**Симптоми**	**%**	**Без симптом**	**%**	**Симптоми невідомі**	**%**
Ч	4095	61,8%	3087	62,2%	96	72,2%	912	59,5%
Ж	2530	38,2%	1,887	37,8%	37	27,8%	620	40,5%

Таблиця 1.2 – Статистика смертності за статтю в США на 14 квітня

1.5 Поширення епідемії по країнам світу

Перший випадок захворювання за межами Китаю зафіксували в Таїланді. Далі вірус підтвердили в: Японії, Південної Кореї, Тайвань, США (рис.1.9).

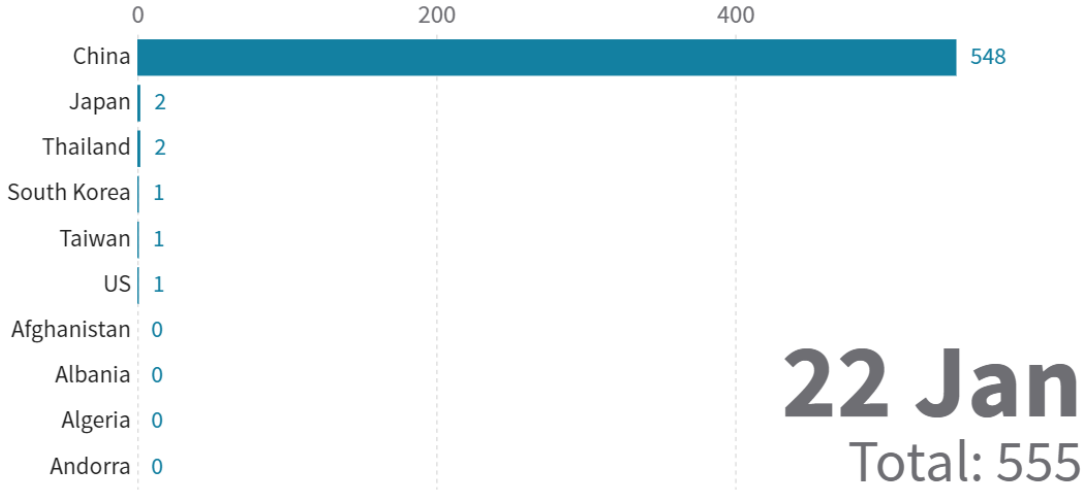


Рисунок 1.9 – Поширення COVID-19 станом на 22.01.2020

Вірус почав поширюватись по всім материкам. Випадків зараження ставало все більше (рис. 1.10).

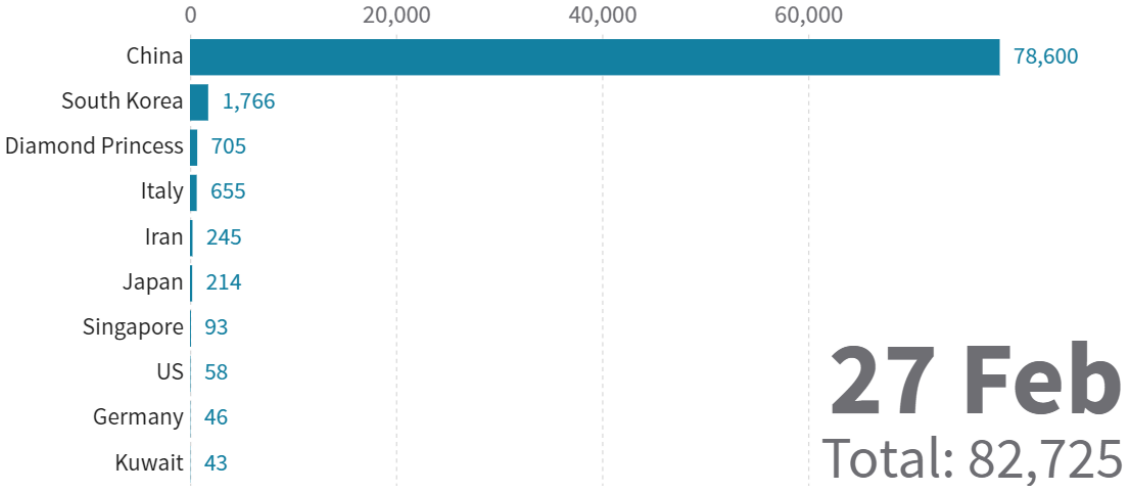


Рисунок 1.10 – Поширення COVID-19 станом на 27.02.2020

В березні Китай досягає піку швидкості зараження і йде на спад. В цей час Європа та Америка тільки набирає обертів нових випадків (рис. 1.11).

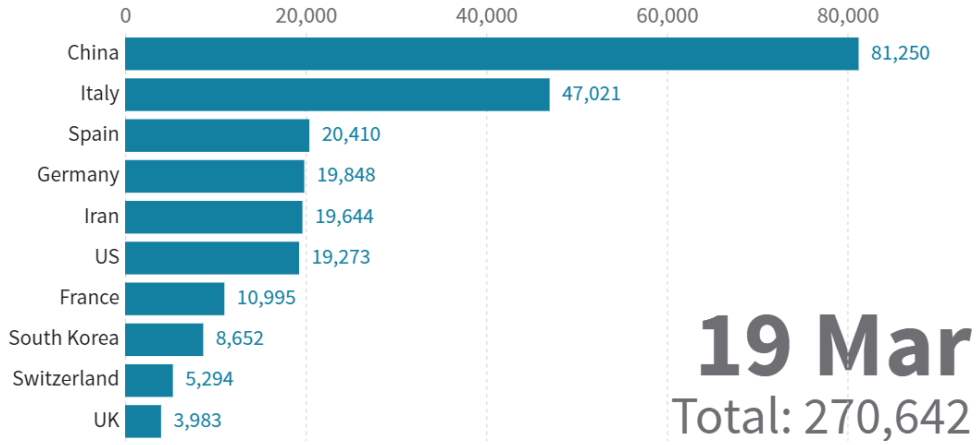


Рисунок 1.11 – Поширення COVID-19 станом на 19.03.2020

Початок травня характеризується тим, що з стрімким зростом нових випадків стає на перше місце США. В Європі ситуація стає все гірше (рис. 1.12)

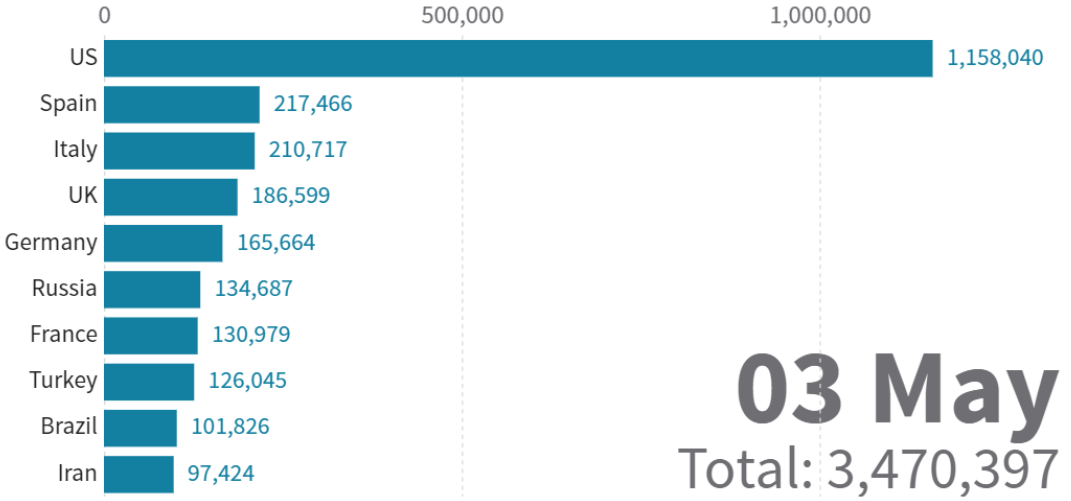


Рисунок 1.12 – Поширення COVID-19 станом на 03.05.2020

В Іспанії та Італії кількість нових випадків в день спадає, а в США, Росії, Бразилії та Британії навпаки зростають (рис. 1.13).

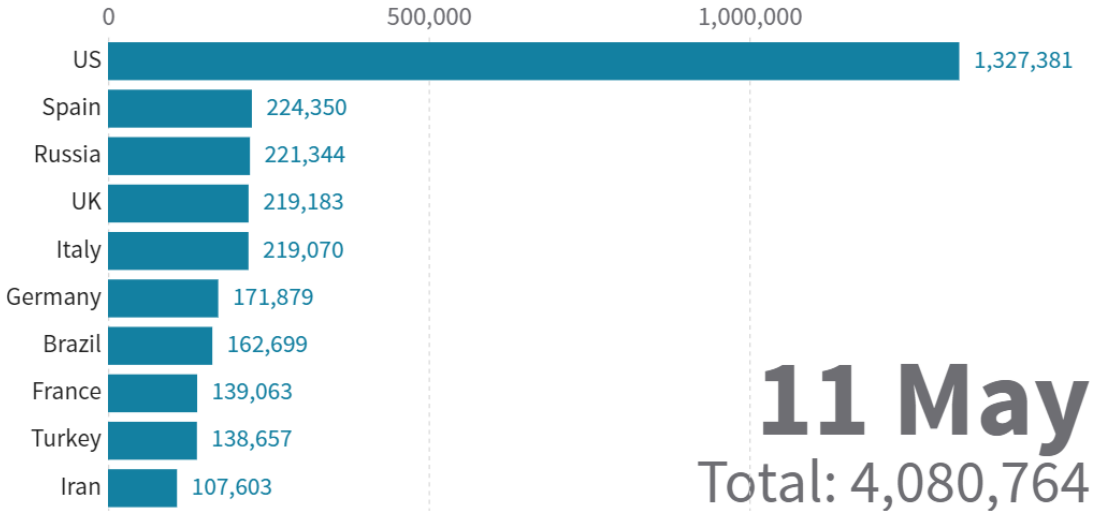


Рисунок 1.13 – Поширення COVID-19 станом на 11.05.2020

На рисунку 1.14 можна відслідкувати поширення епідемії за кількістю випадків в країнах Світу.

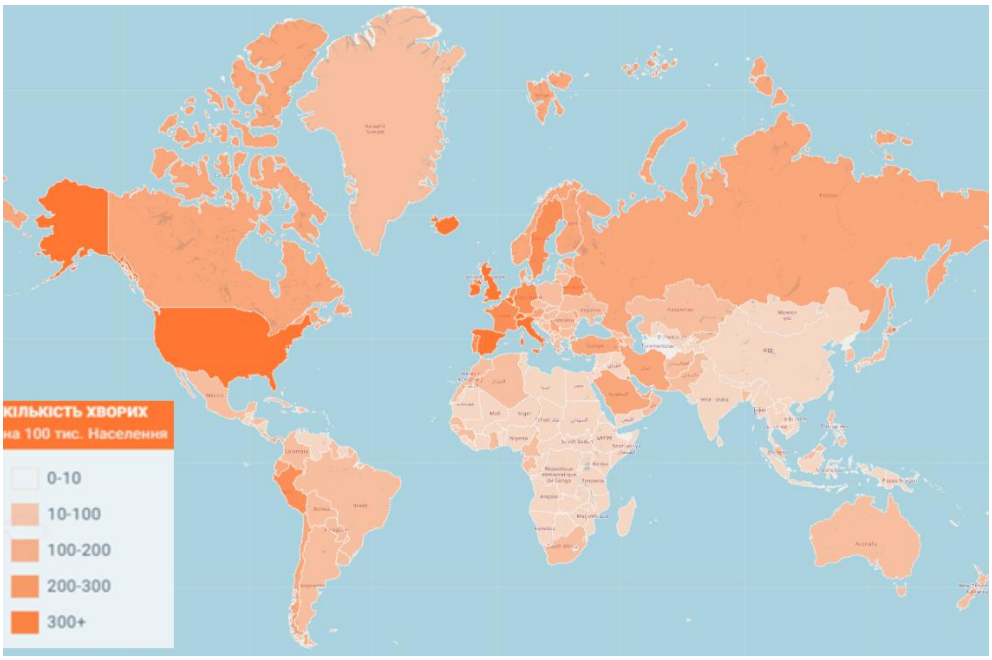


Рисунок 1.14 – Кількість хворих COVID-19 у Світі станом на 11.05.20р.

1.6 Карантинні міри введені в різних країнах Світу відносно COVID-19.

Через поширення коронавірусу по всьому світу, глави держав почали приймати міри, для уникнення масового захворювання. Майже в усіх країнах світу було введено карантинні міри, для прикладу у цьому підрозділі будуть розглянути превентивні міри, введені в країнах з різним фінансовим достатком та рівнем життя.

1.6.1 Україна

Перший випадок захворювання: 3 березня 2020 року.

Першими запобіжними мірами дій в країні було прийняте 25 лютого рішення, щодо перевірки температури усіх прибуваючих на територію країни громадян.

На далі, 11 березня був оголошений Всеукраїнський карантин, який починається 12 березня та мав закінчитися 3 квітня.

Було заборонено:

- Відвідування закладів освіти;
- Проведення всіх масових заходів, у яких бере участь понад 200 осіб, спортивні заходи були дозволені без глядачів.

16 березня був закритий кордон для іноземних громадян. 17 березня, на 14 днів було закрито міжнародне авіаційне, залізничне й автобусне сполучення. З 18 березня припинене міжміське сполучення (залізничне, авіаційне, міжобласне) у Києві, Харкові та Дніпрі припинена робота метрополітену. Також у цей час було встановлене обмежування на кількість пасажирів у автобусах та трамваях.

2 квітня було встановлено жорсткий карантин, заборонено відвідування зон відпочинку, кожна людина обов’язково має одягати захисну маску у громадських місцях. Було встановлено покарання у вигляді штрафу до 17 000 грн. або позбавлення волі строком до 8 років, у разі порушення правил карантину.

1.6.2 Китай

Перший випадок захворювання COVID-19 був зареєстрований в китайському місті Ухань 29 грудня 2019 року.

22 січня 2020 року Ухань і всю провінцію Хубей закрили на карантин. Згодом заходи самоізоляції поширилися по всьому Китаю. До середині березня поширення коронавіруса вдалося взяти під контроль і до 20 березня не було зафіксовано жодного нового випадку зараження.

В середині квітня влада Китаю почали послаблювати карантинні заходи – починали працювати торгові центри, перукарні, кафе і ресторани, майже повністю відновлено транспортне сполучення всередині країни.

1.6.3 Росія

Перший випадок захворювання: 31 січня 2020 року.

Початком введення карантинних мір датується 30 березня. В Москві було введено обмеження на користування громадським транспортом, зачинені магазини, зони відпочинку та інше.

На ранок 31 березня всі ці міри були прийняті ще в 26 регіонах країни. Та вже к 3 квітню усі регіони перейшли на режим самоізоляції.

З 13 квітня 2020 року в країні введені пропуски для користування громадським транспортом

З 30 квітня почали вводити обов'язкові правила щодо масок, вони відрізняються по деяким регіонам, але загалом мають один сенс.

1.6.4 Італія

Перший випадок захворювання: 31 січня 2020 року.

1 березня Рада міністрів ухвалила постанову про організацію стримування спалаху. В указі національна територія Італії була розділена на три області:

- Червона зона (у складі муніципалітетів Бертоніко, Касалпустерленго, Кастельгерундо, Кастільоне Д'Адда, Кодоньо, Фомбіо, Малео, Сан-Фіорано, Сомалія та Терранова-де-Пасеріні в Ломбардії та муніципалітету Ве в Венето), де живе все населення знаходиться в карантині;
- Жовта зона (складається з регіонів Ломбардія, Венето та Емілія-Романья), де призупиняються соціальні та спортивні заходи, а школи, театри, клуби та кінотеатри закриваються;
- Решта національної території, де в громадських місцях оголошуються заходи безпеки та запобігання, а у громадському транспорті виконуються спеціальні санітарії.

4 березня уряд Італії призначив закриття всіх шкіл та університетів по всій країні на два тижні, коли країна досягла 100 смертей від спалаху. Того ж дня уряд постановив, що всі спортивні змагання в Італії будуть проводитись за закритими дверима до 3 квітня.

20 березня Міністерство охорони здоров'я розпорядилось посилити правила щодо вільного руху. Нові заходи забороняли заняття спортом та бігом на відкритому повітрі, за винятком індивідуальних дій та в безпосередній близькості від місця проживання. Парки, дитячі майданчики та громадські зелені були закриті. Крім того, переміщення по всій країні було ще більше обмежено, забороняючи "будь-який рух до резиденції, відмінної від основної", включаючи будинки відпочинку, у вихідні та святкові дні.

1 квітня уряд продовжив термін відключення до 13 квітня, при цьому міністр охорони здоров'я Сперанца заявив, що обмежувальні заходи почали давати перші позитивні результати.

1.6.5 Нігерія

Перший випадок захворювання: 27 лютого 2020 року.

На початку березня міністр охорони здоров'я Нігерії Осагі Еханіре оголосив, що 60 осіб, які контактували з індексним італійським пацієнтом, перебувають у ізоляції, 40 осіб у штаті Огун та 20 у штаті Лагос.

Починаючи с 19 березня уряди різних штатів країни почали закривати учбові заклади, зачинені аеропорти.

Починаючи с 1 квітня штати почали забороняти зібрання та інші масові заходи.

1.6.6 Німеччина

Перший випадок захворювання: 25 лютого 2020 року

У Німеччині є спільний Національний план пандемії , який описує обов'язки та заходи суб’єктів системи охорони здоров’я у разі величезної епідемії. Контроль за епідеміями здійснюють як федеральні органи влади, такі як Інститут Роберта Коха, так і німецькі штати. У німецьких штатів є свої епідемічні плани. На початку березня національний план було розширено щодо боротьби з пандемією коронавірусу, що триває. У цей план включені чотири основні цілі:

- Зниження захворюваності та смертності;
- Забезпечити лікування заражених осіб;
- Обслуговування важливих державних послуг;
- Надійна та точна інформація для осіб, які приймають рішення, медичних працівників, ЗМІ та громадськості;
- План має три етапи, які можуть згодом перекриватися через регіональні відмінності в еволюції пандемії;
- Затримання (обставини виділених справ та кластерів);
- Захист (обставини подальшого поширення інфекції та невідомі джерела інфекції);
- Пом'якшення наслідків (обставини поширених інфекцій).

На стадії стримування органи охорони здоров’я зосереджуються на виявленні контактних осіб, які розміщені в особистому карантині та підлягають моніторингу та тестуванню. Особистий карантин контролюють місцеві медичні установи. Роблячи це, влада намагається утримати ланцюги інфекції короткими, що призведе до скорочення скупчень. На етапі захисту стратегія зміниться на використання прямих заходів щодо захисту вразливих осіб від зараження. Етап пом'якшення в кінцевому підсумку спробувати уникнути шипів інтенсивного лікування з метою підтримки медичних послуг.

1.7 Висновки

У першому розділі розглянуто історію виникнення COVID-19, досліджено методи тестування COVID-19 у людини та фактори, які впливають на протікання хвороби. Проведено аналіз поширення епідемії за різними країнам світу та карантинні міри введені в різних країнах світу відносно COVID-19.

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Інститут прикладного системного аналізу
Кафедра математичних методів системного аналізу

До захисту допущено:
В.о. завідувача кафедри
_____ Оксана ТИМОЩУК
«___»_____20__ р.

Дипломна робота

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Системи та методи штучного інтелекту»
спеціальності 122 «Комп'ютерні науки та інформаційні технології»
на тему: «Моделі та методи інтелектуального аналізу даних COVID-19. Прогнозування діагнозу та виявлення факторів, які впливають на перебіг захворювання»

Виконав:
студент IV курсу, групи КА-66
Сапєльніков Олександр Сергійович _____

Консультант з економічного розділу:
доцент Шевчук Олена Анатоліївна _____

Рецензент:
доцент, к.т.н. кафедри СП ІПСА,
Безносик Олександр Юрійович _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2020 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Інститут прикладного системного аналізу
Кафедра математичних методів системного аналізу

Рівень вищої освіти – перший (бакалаврський)
Спеціальність – 122 «Комп'ютерні науки та інформаційні технології»
Освітньо-професійна програма «Системи та методи штучного інтелекту»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ Оксана ТИМОЩУК
«25» травня 2020 р.

ЗАВДАННЯ
на дипломну роботу студенту
Сапєльнікову Олександру Сергійовичу

1. Тема роботи «Моделі та методи інтелектуального аналізу даних COVID-19. Прогнозування діагнозу та виявлення факторів, які впливають на перебіг захворювання», керівник роботи доцент кафедри ММСА, Недашківська Надія Іванівна, затверджені наказом по університету від «25» травня 2020 р. № 1143-с

2. Термін подання студентом роботи 8.06.2020.

3. Вихідні дані до роботи : вибірка рентген зображень здорової та хворої на коронавірус людини.

4. Зміст роботи: аналітичний та історичний огляд рентген зображень, теоретичний огляд побудови згортковий нейронної мережи, прогнозування наявності захворювання, функціонально-вартісний аналіз програмного продукту.

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо) рентген зображення, архітектури згортових нейронних

мереж, принцип роботи програмного продукту, результати прогнозування моделей.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		зав дання видав	зав дання прийняв
Економічний	Шевчук О.А.		

7. Дата видачі завдання 27 березня 2020

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1.	Вивчення літератури за темою роботи.	13.04.2020	
2.	Підготовка першого розділу.	21.04.2020	
3.	Підготовка другого розділу.	1.05.2020	
4.	Розробка програмного продукту.	19.05.2020	
5.	Підготовка третього розділу	25.05.2020	
6.	Підготовка економічної частини	28.05.2020	
7.	Оформлення розділів відповідно до нормоконтролю.		
8.	Підготовка презентації доповіді.	30.05.2020	
9.	Оформлення дипломної роботи.		

Студент Олександр САПЄЛЬНІКОВ

Керівник Надія НЕДАШКІВСЬКА

ВСТУП

8 листопада 1895 року фізик Вільгельм Конрад Рентген (1845-1923) стає першою людиною, яка спостерігала рентгенівські промені, значне наукове досягнення, яке в кінцевому рахунку принесло б користь різноманітним галузям, найбільше медицині, зробивши невидиме видимим.

Відкриття Рентгена сталося випадково в його лабораторії Вюрцбург, Німеччина, де він перевіряв, чи можуть катодні промені проходити крізь скло, коли він помітив світіння, що надходить із сусіднього екрану з хімічним покриттям. Він охрестив промені, які спричинили це світіння рентгенівськими променями через їх невідому природу.

Рентгенівські промені - це електромагнітні енергетичні хвилі, які діють аналогічно світловим променам, але на довжинах хвиль приблизно в 1000 разів коротші, ніж у світла. Рентген притулився в своїй лабораторії і провів низку експериментів, щоб краще зрозуміти його відкриття. Він дізнався, що рентгенівські промені проникають у людську плоть, але не речовини більш високої щільності, такі як кістка або свинець, і що їх можна сфотографувати.

Відкриття Рентгена було позначене медичним чудом, і рентген незабаром став важливим діагностичним інструментом у медицині, що дозволило лікарям вперше побачити середину людського тіла без операції. У 1897 р. Рентгенівські промені вперше були використані на військовому полі бою, під час Балканської війни, щоб знайти кулі та зламані кістки всередині пацієнтів.

Нова коронавірусна хвороба людини COVID-19 стала п'ятою документально підтвердженою пандемією з пандемії грипу 1918 року. Спочатку про COVID-19 спочатку повідомлялося в місті Ухань, Китай, а згодом поширилося по всьому світу. Міжнародним комітетом з питань систематики вірусів коронавірус був офіційно названий важким гострим респіраторним синдромом коронавірус 2 (SARS-CoV-2) на основі філогенетичного аналізу. Вважається, що SARS-CoV-2 є розповсюдженням

коронавірусу тварини, а згодом адаптував здатність передачі від людини до людини. Оскільки вірус дуже заразний, він швидко поширюється і постійно розвивається в людській популяції. У цій оглядовій статті ми обговорюємо основні властивості, потенційне походження та еволюцію нового коронавірусу людини. Ці фактори можуть бути вирішальними для досліджень патогенності, противірусних конструкцій та розробки вакцини проти вірусу.

Задача класифікації наявності хвороби, особоливо такої як коронавірус, є дуже важливою, цікавою та актуальною в наш час. На мою думку, завдяки використанню згорткових нейронних мереж, можна досягти успіху в вирішенні цієї задачі. Отже дана бакалаврська робота присвячена дослідженню застосування згортових нейронних мереж задля класифікації рентген знимків хворих на коронавірус та здорових людей.

РОЗДІЛ 2 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

2.1 Особливість предметної області

На разі, 20 травня, коронавірусна хвороба COVID-19 досить має великий потенціал розвитку та поширення, не виключається варіант другої хвилі захворювання а також що ця інфекція є сезонною, і якщо її не подолають повністю, то вже в осені ми можемо зустріти нову хвилю. Саме через це, думки багатьох вчених зараз направлені на те, щоб знаходити нові та кращі варіанти виявлення хвороби у людини якомога швидше, враховуючи не надійність існуючих тестів.

Один з таких варіантів виявлення, це розпізнавання ознак хвороби на рентген знімках грудної клітини. Недавні дослідження показали, що рентген, поступаючись КТ в точності діагностики, проте, може зіграти свою роль в приборканні пандемії коронавіруса.

Перші наукові статті з китайського Ухань звеличували здатності КТ виявляти пошкодження легень, що свідчать про інфікування людини вірусом SARS-CoV-2. Цінність рентгенографії була поставлена під сумнів у результатах наукової роботи, проведеної в Південній Кореї: дослідники виявили, що рентген пропустив три чверті легеневих вузлів, пов'язаних з COVID-19, пізніше знайдених за допомогою КТ[1].

Але подальший аналіз показав, що рентгенографія все ж може бути корисна для виявлення COVID-19 за умови, що лікарі будуть знати про властиві їй обмеження. Рентген є недорогим методом діагностики, рентгенівське обладнання – легке в управлінні, і його мобільність дозволяє проводити обстеження пацієнта прямо в лікарняній палаті, не піддаючи персонал радіографічного відділення додаткової загрозу інфікування.

А застосування нових технологій, таких як штучний інтелект (ШІ) може підвищити корисність рентгенографії, автоматизувавши аналіз рентгенівських знімків грудної клітини[1].

2.1.1 Клінічні ознаки на рентгенівських знімках

Рентгенографія грудної клітини (рис. 2.1) може показувати нерівномірні або дифузні асиметричні помутніння легеневої тканини (рис. 2.2), схожі на пневмонію, але викликану іншими коронавірусами, йдеться в статті доктора Джонатана Родрігеса і його колег, опублікованій 23 березня в журналі Clinical Radiology[2].

Тим часом, дослідники з Вашингтонського університету в Сієтлі надали список з п'яти найпоширеніших ознак COVID-19 на рентгенівських знімках. Найбільш поширеним ознакою було двостороннє ретикулярне вузлове затемнення, виявлене в 52% випадків, за яким слідував синдром матового скла в 48% випадків. Приблизно через 72 години після надходження пацієнтів ці показники зростали до 86% для ретикулярних вузлових затемнень і 67% для синдрому матового скла, написали вони у своєму дослідженні, опублікованому 19 березня в JAMA[3].

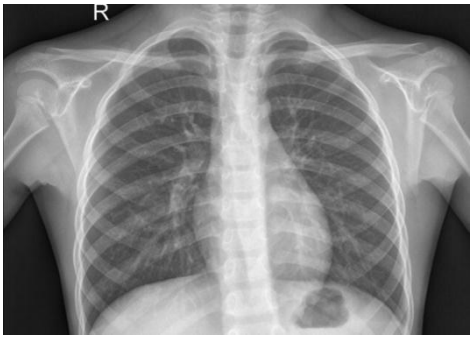


Рисунок 2.1 – рентген знімок здорової людини

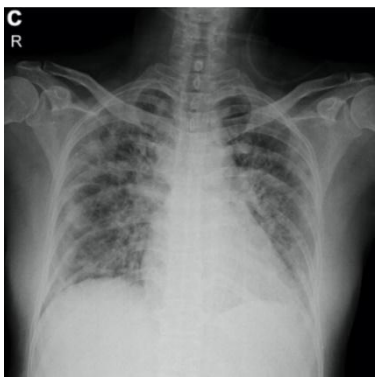


Рисунок 2.2 – рентген знімок людини з ознаками вірусу SARS-CoV-2

2.2 Постанова дослідження

Метою дослідження даної наукової роботи є вивчення та порівняння точності способів класифікації рентген зображень легень, здорової людини та людини хворої на коронавірус. В якості датасетів для дослідження взяті знімки із сайту збірнику датасетів keggle.com.

Розглядаються два датасети, рентген знімки здорової людини, на яких не має бути ознак хвороби і навпаки, знімки хворих людей, на яких завдяки згортковим нейронним мережам будуть знайдені фактори захворювання.

Кінцевою очікуваною метою є реалізація програмному продукту, в основу якого входять різні згорткові моделі нейронних мереж, здатні розпізнавати признаки вірусу, на вхід спочатку подається тренувальна збірка рентген зображень, для навчання моделі, і в наслідку навчання на модель подається тестова вибірка, на виході маємо один з двох класів Non_covid та Covid.

2.3 Порівняльний аналіз методів класифікації зображень

Існує нескінчена кількість різних об’єктів, які можна класифікувати по конкретним категорія (класам) та підкатегоріям. Виділяють більше 100 000 категорій, де близько 3-5% з них є істотними[6]. Процес кластеризації зображень, що є пошуком в них однорідних областей, називається сегментацією. Вважається першим етапом аналізу зображень. Вона базова процедура практично у всіх задачах обробки зображень за допомогою систем комп’ютерного зору. Такі системи застосовуються в промислових роботах нового покоління, телевізійних системах, транспортних роботах, графічних системах і т. д. Як і інші завдання цієї області, сегментація не може бути повністю формалізована, вона включає в себе елементи фільтрації перешкод і виділення зображень. Класифікація методів сегментації описується в різних

роботах, присвячених цьому питанню. Часто їх підрозділяють на ті, які виділяють області однорідної яскравості або кольору, і ті, які визначають однорідності інших властивостей, частіше за все текстури.

Методи першого типу розглядаються вже доволі довго і на їх тему є багато публікації, але останній час більш популярним та досліджуваними стали саме методи 2-го типу.

Інший важливий критерій, по якому можна класифікувати методи сегментації, це характеристики областей. Вони, в одному випадку, можуть бути задані заздалегідь (наприклад, бібліотека еталонів, текстур), а в іншому їх необхідно отримати в процесі сегментації. Якщо область може бути охарактеризована тільки як "фон" або "об'єкт", то сегментація називається грубою, вона частіше використовується при невідомих заздалегідь характеристиках. Якщо характеристик більше, наприклад, "фон", "об'єкт класу 1", "об'єкт класу 2" і т. д., то сегментацію називають розмальовкою[5]. Її часто застосовують при відомих характеристиках. Ці методи можна комбінувати: проводить спочатку грубу сегментацію, а потім багатозначну і остаточну .

Основні існуючі методи сегментації, наводяться на рисунку 2.3. Кореляційні методи застосовуються в разі, якщо відомі еталони об'єктів. Вони ефективні в системах прикладного телебачення і відносяться більше до області розпізнавання зображень. Порогові методи застосовуються при існуванні стабільних відмінностей в яскравості окремих областей. Методи нарощування областей вважаються ефективними за наявності стійкої зв'язності всередині окремих сегментів [1].

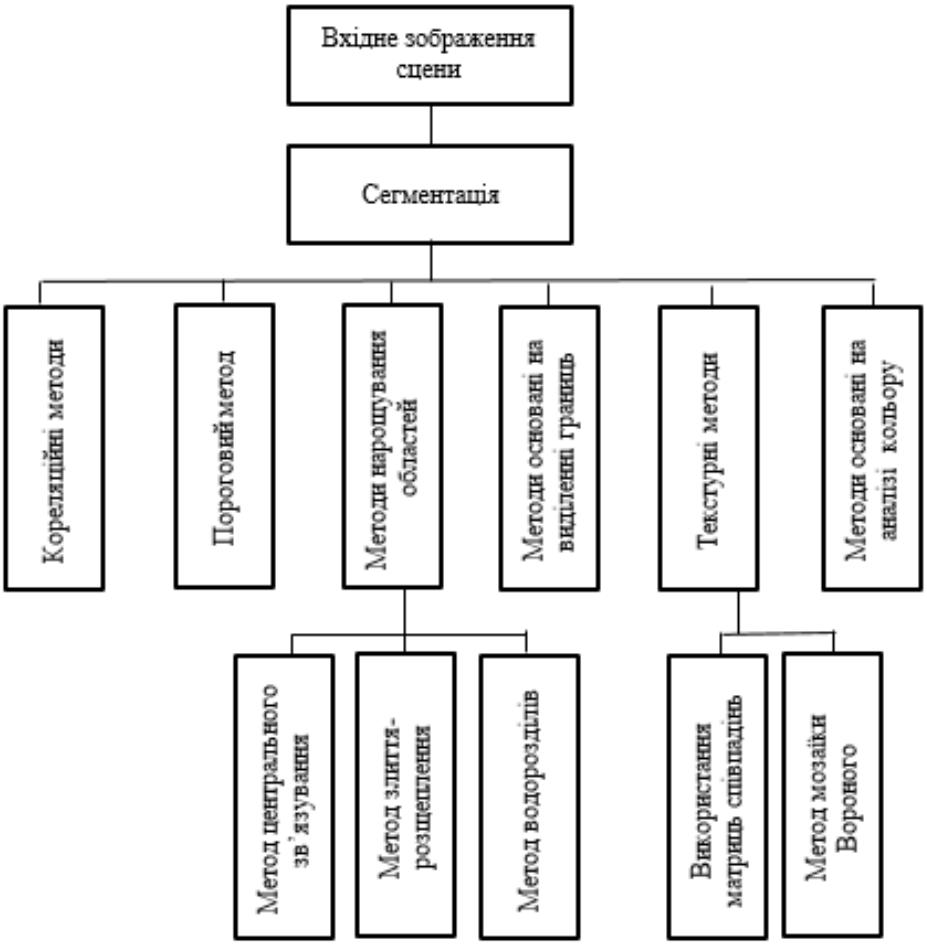


Рисунок 2.3 – Класифікація методів сегментації зображень

Метод виділення границь гарно застосовується, якщо границі зображень чіткі та стабільні. Для опису та сегментації властивостей зображень (однорідності, шорсткості, регулярності) – змінюють текстурні методи, які умовно діляться на дві категорії: статистичні та структурні. Приклад статистичного підходу – це використання матриць спеціального виду, які формуються на основі вихідних зображень; структурного – мозаїка Вороного. Методи, засновані на аналізі кольору, по своїй суті є комбінованими.

2.3.1 Загальна математична модель сегментації

Нехай D(m x n) – растр або область поля зору, на якому задано зображення B(i,j);

$D_k \subset D$ – область k – го об’єкту k = 1, 2, …, s; $D_\varphi \subset D$ – область фону.

Вважаємо:

$$D_1 \cup D_2 \cup \dots D_S \cup D_\varphi = D, D_i \cup D_j = \emptyset, i \neq j \quad (2.1)$$

Розглянемо дискретне зображення $B(i,j), i = 0, \dots, m, j = 0, \dots n$. Зображення є сукупністю зображень окремих об’єктів і фону[6]. Його можна представити у вигляді:

$$B(i,j) = H_1(i,j) + \cdots + H_S(i,j) + H_\varphi(i,j), \quad (2.2)$$

де s – це число об’єктів. $H_k(i,j)$ – зображення k-го об’єкта або видимої його частини, k = 1, 2,…,s; $H_\varphi(i,j)$ – зображення фону. Якщо (i, j) не влучає в область k-го об’єкта, то $H_k(i,j)$ = 0, $H_k(i,j) = B(i,j)$ для $(i,j) \in D_\varphi$.

Задача сегментації першого типу зображення полягає в побудові предиката виду:

$$\pi(i,j) = \begin{cases} k, \text{якщо } (i,j) \in D_k \\ 0, \text{якщо } (i,j) \in D_\varphi \end{cases}$$

Це означає, що кожна точка $(i,j) OD$ зображення $B(i,j)$ отримує смислову мітку з номером $p(i,j)$. Грубо кажучи, в ідеалі крапки з однією міткою утворюють область одного окремого об’єкта, міткою 0 розмічається область фону.

2.3.2 Методи аналізу різниці яскравості

Пороговий метод.

Нехай задано зображення $B(i,j), s = 1$(один об'єкт), яскравість точок знаходиться в межах $[T_1, T_2]$, а яскравість точок фону в ці межі не входять. Якщо $B(i,j) \in [T_1, T_2]$, то точку (i,j) вважаєм належною до області об'єкта, а в іншому випадку – області фону. У випадку $s > 1$ повинні бути відомими відрізки $[T_1^k{}_1, T_2^k]$, в межах яких знаходяться яскравості k-х об'єктів[4]. Ці точки не повинні перетинатись. Розмітка точок відбувається за допомогою відображення.

Виділення границь.

При такому способі сегментації об'єкти представляються їх границями. Граничні точки прийнято рахувати точки різкого перепаду функції яскравості.

Для знаходження граничних точок використовують чисельне диференціювання. Найбільш поширеним методом є градієнтний, відомий як метод контрастування або просторового диференціювання. Застосовуючи маску (фільтр) до зображення, отримують зображення градієнтів. Воно відрізняється від вихідного підкресленими перепадами яскравості[5]. Точка (i,j) належить контуру, якщо яскравість зображення градієнтів перевищує де-який поріг, котрий може визначатись по гістограмам.

Аналіз кольору.

Сегментація шляхом аналізу кольору заснована, по суті, на його впізнанні. Признаками слугують, наприклад, при компоненти (координати) від функції випромінювання $B(1)$ в кожній точці (i,j): $C_1 = \int_\lambda B(\lambda) k_l(\lambda)\, d\lambda, l = 1,2,3$. Спектральні криві чуттєві $K_1(\lambda), K_2(\lambda), K_3(\lambda)$ можуть відповідати функціям трьох видів колбочок ока людини, але на практиці використовується багато інших систем кодування кольорових компонент, включаючи дво- і чотирьох компонентні системи[7].

Метод нарощення областей (Злиття – розщеплення).

Метод складається в розбитті зображення на квадрати де-яким чином. Потім проводиться аналіз однорідності цих квадратів, частіше всього

аналізується однорідність яскравості. Якщо квадрат не задовольняє умові однорідності, то він замінюється чотирма підквадратами. Якщо ж ділянка з чотирьох сусідніх квадратів з'ясовується такою, що для неї виповнюються всі умови однорідності, то ці чотири квадрата об'єднуються в один. Результатом злиття-розщеплення може слугувати певна структура з інформацією про квадрати, скоріш за все – граф, а може бути і зображення, в якому всі пікселі в середині однорідної області мають однакову яскравість.

2.3.3 Текстурні методи

Матриця співпадінь.

Цей метод входить в групу статичних. Шляхом обчислення для кожної ділянки так званої матриці співпадінь, він дозволяє визначити чи мають ділянки зображення текстури одного класу[8].

Розглянемо ділянку $N \times N$. Маємо множину яскравості $\{B(i,j), i = 1, \dots, N, j = 1, \dots N\}$ з G градаціями сірого. Визначається вектор зміщення $d = (dx, dy)$. Вводиться матриця співпадінь $G \times G$, яка позначається як P_d. Елемент (a, b) матриці P_d дорівнює числу випадків, коли від точки з яскравістю a на відстані, визначальним вектором d, знаходиться точка з яскравістю b. Формально це записується як:

$$P_d(a,b) = \sum_{r,S} p\big(a,b,(r,s),(t,v)\big) \tag{2.3}$$

$$p\big(a,b,(r,s),(t,v)\big) = \begin{cases} 1, & \text{якщо } B(r,s) = a, B(t,v) = b \\ 0, & \text{в іншому випадку} \end{cases} \tag{2.4}$$

$$(t, v) = (r + dx, s + dy) \tag{2.5}$$

На основі матриці P_d можуть бути обчислені різні характеристики, наприклад енергія:

$$\sum_a \sum_b P_d^2(a, b) \tag{2.6}$$

Метод мозаїки Вороного.

Сегментація з використанням цього методу включає в себе три кроки: побудова примітивів, складання мозаїки, аналізу елементів мозаїки. Частіше всього для отримання примітивів до зображення застосовують фільтри, к виділення границь.

Потім вибирають точки локальних максимумів, до яких застосовують метод нарощування, в результаті чого отримують компоненти із 8-ми зв’язних елементів[7]. Отримані таким чином компоненти або точки локальних максимумів визначають як примітиви. Потім будують мозаїчне розбиття Вороного для примітивів.

2.4 Висновки

В першому розділі розглянуто об’єкт дослідження. Оглянуто та проаналізовано існуючі методі класифікації зображень. Розглянуто історію виникнення рентген знимків та рентген апарату в цілому. Проаналізувавши інформацію я прийняв рішення, що для виконання цієї задачі найкраще всього скористатися моделями згортових нейронних мереж для класифікації зображень.

РОЗДІЛ 3 МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ ГЛИБОКИХ НЕЙРОННИХ МЕРЕЖ ПРЯМОГО РОЗПОВСЮДЖЕННЯ

3.1 Побудова скритих і вихідних шарів нейронної мережі

3.1.1 Вихідний шар SoftMax

Формально функція softmax визначається наступним чином:

$$\mathrm{softmax}(z_i) = \frac{\exp(z_i)}{\sum_j \exp(z_i)} \tag{3.1}$$

Як і в випадку логістичної сигмоїд, використання функції exp дає гарні результати при навчанні softmax з метою породження вихідного значення y із застосуванням логарифмічної правдоподібності[11]. В цьому випадку ми хочемо максимізувати:

$$\log P(y = i; z) = \log \mathrm{softmax}(z_i) \tag{3.2}$$

Визначення softmax через exp природно, тому що логарифм, що входить в логарифмічна правдоподібність, компенсує потенціювання в softmax:

$$\log \mathrm{softmax}(z_i) = z_i - \log \sum_j \exp(z_j) \tag{3.3}$$

Перший член у виразі (3.3) показує, що вхід z_i дає прямий внесок в функцію вартості. Оскільки цей член не відчуває насичення, то навчання завжди може продовжитися, навіть якщо вклад z_i в другій член стає дуже

малий. При максимізації логарифмічного правдоподібності перший член заохочує збільшення z_i, а другий – зменшення всіх елементів z.

Як і сигмоїда, функція активації softmax схильна до насичення. У сигмоїді всього один вихід, і вона насичується, коли абсолютна величина аргументу дуже велика. У softmax вихідних значень кілька. Вони насичуються, коли велика абсолютна величина різниць між вхідними значеннями.

Щоб зрозуміти, як softmax реагує на різницю між вхідними значеннями, помітимо, що вихід softmax інваріантний щодо додавання одного і того ж скаляра до всіх входів:

$$\text{softmax}(z) = \text{softmax}(z + c). \tag{3.4}$$

Користуючись цією властивістю, ми можемо вивести чисельно стійкий варіант soft-max:

$$\text{softmax}(z) = \text{softmax}(z - \max i\, z_i) \tag{3.5}$$

Цей новий варіант дозволяє обчислювати softmax з малими чисельними похибка ностями, навіть коли z містить дуже великі за абсолютною величиною елементи. Вивчивши чисельно стійкий варіант, ми бачимо, що на функцію softmax впливає відхилення її аргументів від maxi z_i.[11]

3.1.2 Скриті шари ReLU

CNN складається з вхідного шару, вихідного шару, а також множини прихованих шарів. Приховані шари CNN зазвичай складаються з згорткових шарів, об'єднання шарів, повно зв’язних шарів і шарів нормалізації (ReLU).

Для складніших моделей можна використовувати додаткові шари.

Архітектура CNN показала відмінні результати у багатьох проблемах із комп’ютерним зором та машинним навчанням. Ця модель CNN широко використовується в сучасних програмах машинного навчання завдяки постійній ефективності рекордів. Множення матричного вектора лежить в основі представлення даних і ваг. Кожен з шарів містить різний набір характеристик для набору зображень. Наприклад, якщо зображення обличчя є входом до CNN, мережа вивчить деякі основні характеристики, такі як краї, яскраві плями, темні плями, фігури тощо, у своїх початкових шарах. Наступний набір шарів буде складатися з фігур і предметів, пов'язаних із зображенням, які можна розпізнати, такі як: очі, ніс і рот[12]. Наступний шар складається з аспектів, схожих на фактичні обличчя, іншими словами, форми та об'єкти, які мережа може використовувати для визначення людського обличчя. CNN відповідає деталям, а не цілому зображенню, тому розбиває процес класифікації зображення на більш дрібні частини (характеристики). Сітка 3x3 визначена для відображення функції вилучення CNN для оцінки.

Наступний процес, відомий як фільтрування, полягає в тому, що один за одним кожен піксель множиться на відповідний піксель функції, і після завершення всі значення підсумовуються та діляться на загальну кількість пікселів у просторі функцій. Кінцеве значення для функції потім розміщується в патчі функції. Цей процес повторюється для інших патчів функцій з подальшим випробуванням усіх можливих повторних застосувань цього фільтра, який відомий як згортка. Наступний шар CNN називається "максимальним об'єднанням", що передбачає зменшення стека зображення. Щоб об'єднати зображення, потрібно визначити розмір вікна (наприклад, зазвичай 2x2 / 3x3 пікселі), також слід визначити крок (наприклад, 2 пікселі). Потім вікно фільтрується через зображення кроками, при цьому максимальне значення записується для кожного вікна. Максимальне об'єднання зменшує розмірність кожної карти 40 зображень, зберігаючи найважливішу інформацію. Нормалізаційний шар CNN, який також називають процесом

випрямленого лінійного блоку (ReLU), включає зміну всіх негативних значень у відфільтрованому зображенні на 0. Цей крок повторюється на всіх відфільтрованих зображеннях, шар ReLU збільшує нелінійні властивості моделі[14].

Наступним кроком CNN є складання шарів (згортка, об'єднання, ReLU), так що вихід одного шару стає входом наступного. Заключний шар в архітектурі CNN називається повно зв’язним шаром, також відомим як класифікатор. У цьому шарі кожне значення отримує голос за визначення класифікації зображення. Кожен додатковий шар дозволяє мережі вивчити ще більш складні комбінації функцій для кращого прийняття рішень. Значення, що використовуються для шару згортки, а також ваги для повно зв’язних шарів отримують за допомогою зворотного розповсюдження, що здійснюється глибокою нейронною мережею. Зворотне розповсюдження полягає в тому, що нейронна мережа використовує помилку в остаточній відповіді, щоб визначити, наскільки мережа коригується та змінюється.

3.2 Регулярізація Dropout

Однією з проблем глибоких нейронних мереж (Deep Neural Networks) є перенавчання (overfitting), яке полягає в тому, що модель добре описує тільки елементи навчальної вибірки, адаптується до навчальних прикладів занадто добре [12]. В цьому випадку прогнозна здатність моделі є низькою: модель не вміє класифікувати приклади, які не брали участі в навчанні. За останні роки було запропоновано декілька рішень проблеми перенавчання, одне з них – це Dropout (дропаут).

Графічне представлення методу Dropout на рисунку 3.1.

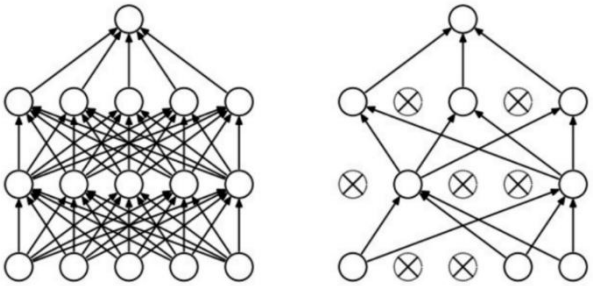


Рисунок 3.1 – До і після Dropout

Графічне представлення методу Dropout рисунок 3.1. Зліва – нейронна мережа до того, як до неї застосували дропаут, а праворуч – мережа після застосування дропауту.

Мережа, зображена зліва, використовується при тестуванні, після того як проведено навчання її параметрів.

Головна ідея дропауту полягає в тому, що замість навчання однієї мережі виконується навчання ансамблю декількох мереж; отримані результати потім усереднюють.

Мережі для навчання будуються шляхом виключення з мережі нейронів з ймовірністю p, так що імовірність того, що нейрон залишиться в мережі, становить $q = 1 - p$. Звідси і назва методу. Виключення нейрона означає, що він повертає 0 при будь-яких вхідних даних та параметрах.

На кожному з етапів алгоритму зворотного поширення помилки виключені нейрони не вносять свого внеску в процес навчання. У зв'язку з цим виключення хоча б одного нейрону рівносильно навчанню нової нейронної мережі [12].

Dropout вимикає нейрони з ймовірністю, як наслідок, залишає їх включеними з ймовірністю $\mathrm{q} = 1 - \mathrm{p}$.

Імовірність виключення кожного нейрона однакова. Це означає наступне:

За умови, що:

- $\mathrm{h(x)} = \mathrm{xW} + \mathrm{b}$ – лінійна проекція вхідного d(i)-мірного вектора x на d(h) мірний простір вихідних значень;

- $a(h)$ – функція активації.

Застосування дропауту до даної мережі на етапі навчання можна представити за допомогою зміненої функції активації:

.

$$F(h) = D \odot a(h), \tag{3.6}$$

де $D = (x_1, \dots, x_{dh}) - d_h - 1$ – мірний вектор випадкових велечин X_i розподіленний за законом Бернуллі X_i має наступний розподіл ймовірностей:

$$f(k; p) = \begin{cases} p, якщо\ k = 1 \\ 1 - p, якщо\ k = 0' \end{cases} \tag{3.7}$$

де k – це усі можливі вихідні значення.

Очевидно, що ця випадкова величина ідеально відповідає Dropout, застосованого до одного нейрона. Дійсно, нейрон вимикають з ймовірністю $p = P(k = 1)$, в іншому випадку – залишають включеним.

Подивимося на застосування Dropout до **i**-го нейрона:

$$O_i = X_i a\left(\sum_{k=1}^{d_i} w_k x_k + b\right) = \begin{cases} a\left(\sum_{k=1}^{d_i} w_k x_k + b\right), якщо\ X_i = 1, \\ 0, якщо\ X_i = 0 \end{cases} \tag{3.8}$$

де $P(X_i = 0) = p$.

Так як на етапі навчання нейрон з ймовірністю q залишається в мережі, то на етапі тестування виконують емулювання поведінки ансамблю нейронних мереж, який було використано при навчанні [12]. Для цього під час тестування мережі функцію активації множать на коефіцієнт q.

Таким чином, під час навчання:

$$O_i = X_i a\left(\sum_{k=1}^{d_i} w_k x_k + b\right) \quad (3.9)$$

Під час тестування:

$$O_i = qa\left(\sum_{k=1}^{d_i} w_k x_k + b\right) \quad (3.10)$$

3.3 Алгоритми оптимізації глибоких нейронних мереж

3.3.1 Функція втрат Кульбака-Лейблера

Розбіжність Кульбака-Лейблера визначається формулою:

$$D_{KL}(\hat{p}_{data}||p_{model}) = E_{x-\hat{p}_{data}}[log\hat{p}_{data}(x) - logp_{model}(x)] \quad (3.11)$$

Перший член різниці в квадратних дужках залежить тільки від породжує дані процесу, але не від моделі[13]. Отже, при навчанні моделі, мінімізуючи розбіжність КЛ, ми повинні мінімізувати тільки величину:

$$D_{KL}(\hat{p}_{data}||p_{model}) = E_{x-\hat{p}_{data}}[log\hat{p}_{data}(x) - logp_{model}(x)] \quad (3.12)$$

$$-E_{x-\hat{p}_{data}}[logp_{model}(x)] \quad (3.13)$$

Мінімізація розбіжності КЛ в точності відповідає мінімізації перехресною ентропії між розподілами. Багато авторів вживають термін «перехресна ентропія» для позначення виключно негативного логарифмічної правдоподібності розподілу Бернуллі або softmax, але це неправильно. Будь-яка функція втрат, що містить негативне логарифмічна правдоподібність, є перехресної ентропією між емпіричним розподілом, визначеним навчальним набором, і розподілом, визначеним моделлю.

Наприклад, середньоквадратична помилка – перехресна ентропія між емпіричним розподілом і гаусом моделлю.

3.3.2 Стохастичний градієнтний спуск

Градієнтні методи представляють клас алгоритмів оптимізації, які використовують в машинному навчанні та інших прикладних областях [14]. В даній роботі градієнтний метод розглядається як спосіб для підбору векторів синаптичних ваг w в лінійному класифікаторі. Нехай $y^* : X \to Y$ - цільова змінна, відома лише на елементах навчальної вибірки:

$$X^l = (x_i, y_i)^l_{i=1,} y_i = y^*(x_i) \quad (3.14)$$

Алгоритм a(x,w), що апроксимує залежність y* у випадку лінійного класифікатора, має вигляд:

$$a(x, w) = \varphi(\sum_{i=1}^{n} w_j x^j - w_0) \quad (3.15)$$

де $\varphi(z)$ – функція активації. В найпростішому випадку покладають $\varphi(z)=sign(z)$

Вхідні дані для алгоритма стохастичного градієнта наступні:

X^l - навчальна вибірка

η - темп навчання

λ - параметр згладжування функціоналу Q

Результатом цього алгоритма є вектор ваг w.

Етапи алгоритма стохастичного градієнта [22]:

1.Ініціалізувати ваги w_j $j = 0, .., n$.

2.Ініціалізувати поточну оцінку функціоналу:

$$Q := \sum_{i=1}^{l} L(a(x_i, w), y_i) \quad (3.16)$$

3.Повторювати:

Вибрати об'єкт x_i із X^l (наприклад, випадковим чином);

Обчислити вихідне значення алгоритму $a(x_i, w)$ та помилку:

$$\varepsilon_i := L(a(x_i, w),\ y_i); \quad (3.17)$$

Зробити крок градієнтного спуску:

$$w := w\ -\ \eta L'_a(a(x_i, w),\ y_i)\varphi'(< w, x_i >)x_i \quad (3.18)$$

Оцінити значення функціоналу:

$$Q := (1\ -\ \lambda)Q\ +\ \lambda\varepsilon_i \quad (3.19)$$

Поки значення Q не стабілізується та/або ваги w не припинять змінюватись.

3.3.3 Метод Адам

Метод Adam – один з алгоритмів оптимізації з адаптивною швидкістю навчання. Назва «Adam» – скорочення від «adaptive moments», що означає адаптивні моменти. Цей алгоритм розглядають як комбінацію алгоритму RMSProp та імпульсного методу, що має декілька відмінностей.

По-перше, імпульс в алгоритмі Adam включено безпосередньо, а саме, у вигляді оцінки перших кроків з експонентними вагами градієнта [16]. Найпряміший спосіб додати імпульс в RMSProp – це застосувати його до

масштабованих градієнтів. У використанні імпульсу в поєднанні з масштабуванням, проте, поки немає ясного теоретичного обґрунтування.

По-друге, алгоритм Adam включає поправку на зміщення в оцінки як перших моментів, це член імпульсу, так і других, тобто нецентрованих моментів, для врахування їх ініціалізації на початку координат [16]. Алгоритм RMSProp також включає оцінку для нецентрованого другого моменту, проте в ній немає поправочного коефіцієнту. Таким чином, на відміну від алгоритму Adam, в RMSProp оцінка другого моменту може мати велику величину зміщення на ранніх стадіях навчання. Загалом, алгоритм Adam вважається досить стійким до вибору значень гіперпараметрів, хоча швидкість навчання іноді потрібно брати відмінною від пропонованої по умовчанню [22].

Вхідні дані для алгоритму Adam:

- величина кроку ε (за замовчуванням 0.001).

- коефіцієнти експоненціального затухання для оцінок моментів ρ1 і ρ2, що належать діапазону [0, 1) (за замовчуванням 0.9 і 0.999 відповідно).

- невелика константа δ для забезпечення чисельної стійкості.

- початкові значення параметрів θ.

Алгоритм Adam складається з наступних етапів:

Ініціалізувати змінні для першого і другого моментів s = 0, r = 0

Ініціалізувати крок за часом t = 0

while критерій зупинки не виконано do

Вибрати з навчального набору міні-пакет m прикладів {x (1), ..., x (m)} і мітки y (i).

Обчислити градієнт: g ← (1 / m) ∇θΣi L (f (x (i); θ), y (i)).

t ← t + 1

Оновити зміщену оцінку першого моменту: s ← ρ1s + (1 – ρ1) g Оновити зміщену оцінку другого моменту: r ← ρ2r + (1 – ρ2) g ⊙ g

Скорегувати зміщення першого моменту:

$$\hat{\boldsymbol{s}} \leftarrow \frac{\boldsymbol{s}}{1-\rho_1^t} \quad (3.20)$$

Скорегувати зміщення другого моменту:

$$\hat{\boldsymbol{r}} \leftarrow \frac{\boldsymbol{r}}{1-\rho_2^t} \quad (3.21)$$

Обчислити оновлення:

$$\Delta\boldsymbol{\theta} = -\varepsilon \frac{\hat{\boldsymbol{s}}}{\sqrt{\hat{\boldsymbol{r}}} + \delta} \quad (3.22)$$

Застосувати оновлення: θ ← θ + Δθ. end while.

Метод Adam перетворює градієнт наступним чином:

$$S_t := \alpha \cdot S_{t-1} + (1-\alpha) \cdot \nabla E_t^2 \; ; \; S_0 := 0 \quad (3.23)$$

$$D_t := \beta \cdot D_{t-1} + (1-\beta) \cdot \nabla E_t \; ; \; D_0 := 0 \quad (3.24)$$

$$g_t := \frac{D_t}{1-\beta} \cdot \sqrt{\frac{1-\alpha}{S_t}} \quad (3.25)$$

$$\Delta W_t := \eta \cdot \left(g_t + \rho \cdot W_{t-1}\right) + \mu \cdot \Delta W_{t-1} \quad (3.26)$$

де η – коефіцієнт швидкості навчання,

∇E – градієнт функції втрати,

μ – величина моменту,

ΔW_{t-1} – величина зміни ваг порівняно з попередньою ітерацією,

ρ – коефіцієнт регуляризації,

W_{t-1} – значення ваг на попередній ітерації,

$\alpha = 0.999$, $\beta = 0.9$

3.3.4 Ініціалізація ваг методами Глоро і Хе

Достовірно відомо, що нейронна мережа чутлива до початкової ініціалізації параметрів. Основним результатом їх роботи став простий алгоритм ініціалізації, який покращив як швидкість, так і якість навчання та отримав назву-ініціалізація Глоро (Xavier/Glorot initialization)[13].

Розглянемо значення одного нейрону (до застосування функції активації):

$$y = w^{\top} x + b = \sum_i w_i x_i + b, \tag{3.27}$$

де x – вектор вхідних значень, w – вектор параметрів. Таким чином, дисперсія V ar(y) не залежить від члена b, а залежить тільки від вектора вхідних значень і вектора параметрів. Позначимо і-й член суми як $y_i = w_i x_i$.

Припустимо, що x_i та w_i – незалежні, маємо:

$$\begin{aligned} &\mathrm{V\,ar}(y_i) = \mathrm{V\,ar}(w_i x_i) = \mathrm{E}[w_i x_i] - (\mathrm{M}[w_i x_i])^2 \qquad (3 \\ &= \mathrm{M}[x_i]^2\,\mathrm{V\,ar}(w_i) + \mathrm{M}[w_i]^2 \mathrm{V\,ar}(x_i) + \mathrm{V\,ar}(x_i)\mathrm{V\,ar}(w_i)\,, \qquad .28) \end{aligned}$$

де $\mathrm{M}[\cdot]$ – оператор взяття математичного сподівання.

Перевагою цих методів є простота реалізації і відносно висока якість ініціалізації, проте ми не можемо закласти апріорні знання в модель, так як при ініціалізації ці знання не враховуються.

Результати роботи обох методів зображено на (Рисунок 3.2) та (Рисунок 3.3)

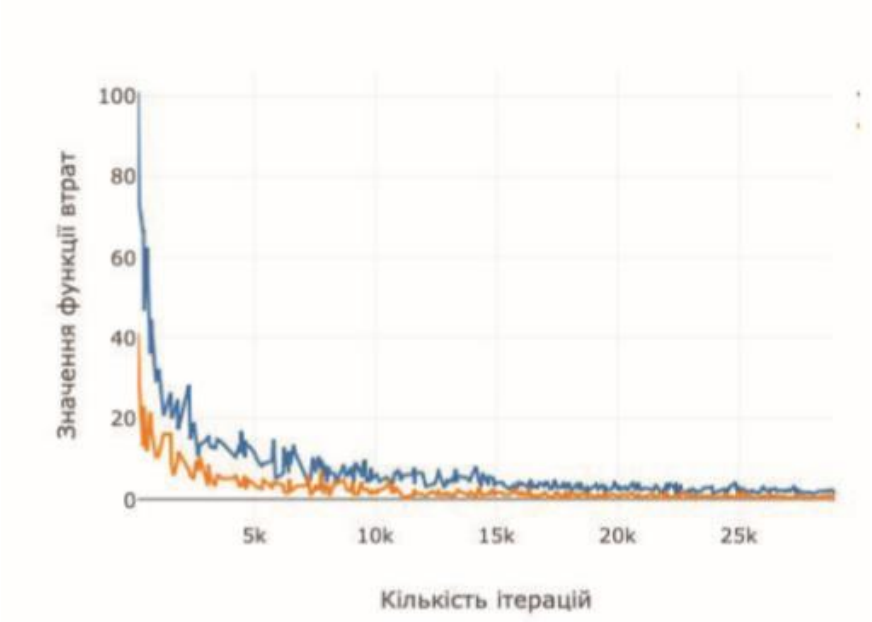


Рисунок 3.2 – Порівняння навчання нейронної мережі на базі цифр MNIST з використанням ініціалізації Глоро

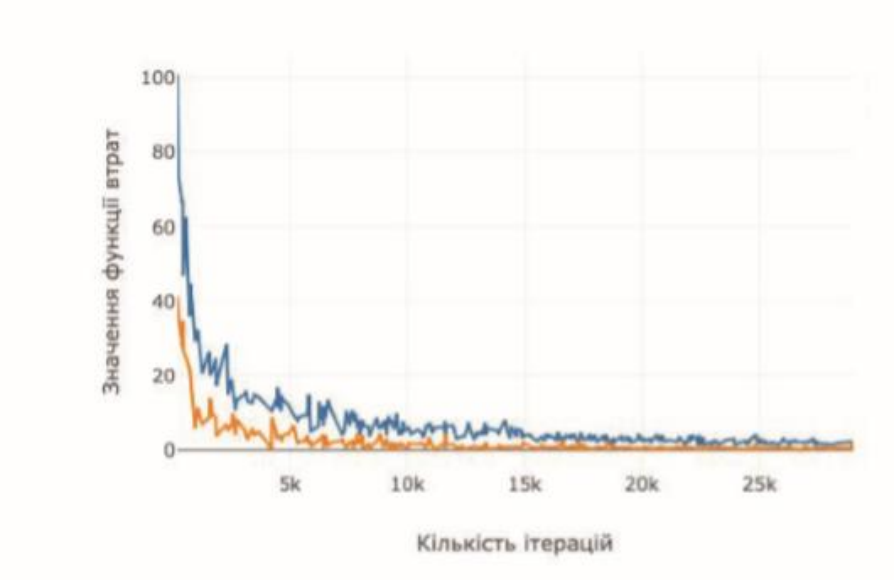


Рисунок 3.3 – Порівняння навчання нейронної мережі на базі цифр MNIST з використанням ініціалізації Хе.

3.4 Згорткові нейронні мережі

Згорткові нейронні мережі (Convolitional Neiral Network – CNN) з'явилися в результаті вивчення зорової кори головного мозку і застосовувалися в розпізнаванні зображень, починаючи з 1980-х років[16]. Завдяки зростанню обчислювальної потужності в останні кілька років, збільшення обсягу доступних навчальних даних і появи трюків для навчання глибоких мереж CNN вдалося досягти надлюдської продуктивності при вирішенні ряду складних зорових завдань. Вони приводять в дію потужні служби пошуку зображень, безпілотні автомобілі, системи автоматичної класифікації відеороликів і т.п. Крім того, мережі CNN не обмежуються зоровим сприйняттям: вони також успішно вирішують інші завдання на зразок розпізнавання мови (voice recognition) або обробки природної мови (Natural Language Processing (NLP))

Найважливіший будівельний блок мережі CNN це згортковий шар : нейрони в першому згорткованому шарі не пов'язані з кожним поодиноким пікселем у вхідному зображенні , а тільки з пікселями в власних рецепторних полях. У свою чергу кожен нейрон у другому згорткованому шарі пов'язаний тільки з нейронами, що знаходяться всередині невеликого прямокутника в першому шарі. Така архітектура дозволяє мережі зосередитися на низькорівневих ознаках в першому прихованому шарі, потім скомпонувати їх в ознаки більш високого рівня в наступному прихованому шарі і т.д. Подібна ієрархічна структура поширена в реальних зображеннях, що і є однією з причин, чому мережі CNN настільки добре працюють при розпізнаванні зображень.

3.4.1 Зв’язування і розділення параметрів

Якщо розглянути традиційну нейронну мережу, то в ній кожний вихідний блок взаємодіє з кожним вхідним. В той час як в згортових мережах взаємодія є розрідженою, внаслідок того, що ядро має меншу розмірність ніж вхід. Це можна інтерпретувати як: обробка частини зображення буде

відбуватися незалежно від того де вона розташована[16]. Завдяки цьому можна зберігати менше параметрів, отже знижаються вимого щодо об’єму пам’яті та для обчислення вихідного сигналу треба не так багато операцій, отже збільшується ефективність.

На Рисунку 3.4 зверху зображено шар s, утворений згорткою з ядром розмірності 3, отже x3 впливає лише на 3 виходи, в той же час на знизу на цьому зображенні шар s утворено множенням на матрицю, як в традиційній нейронній мережі, зв’язність не є розрідженою.

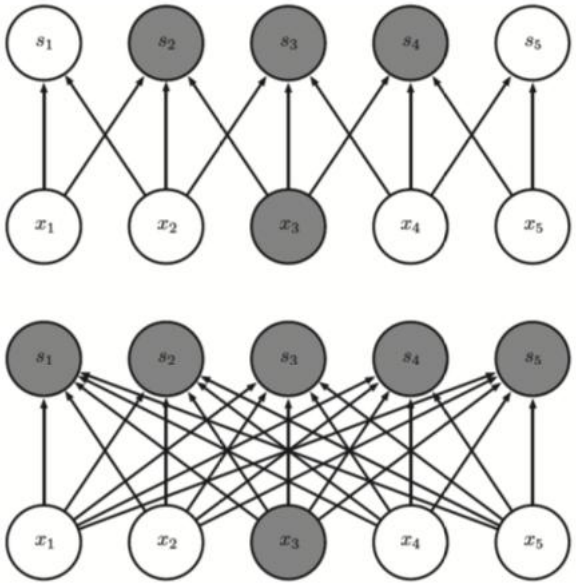


Рисунок 3.4 – шляхи утворення шару s

Мережа може якісно описувати складні взаємодії між багатьма змінними, якщо виконується важлива умова: вона складається з простих елементів, де кожний з них описує тільки розріджену взаємодію[15]. На Рисунку 3.5 наведено приклад опосередкованої взаємодії блоків нижнього рівня з більшою частиною вхідного блоку.

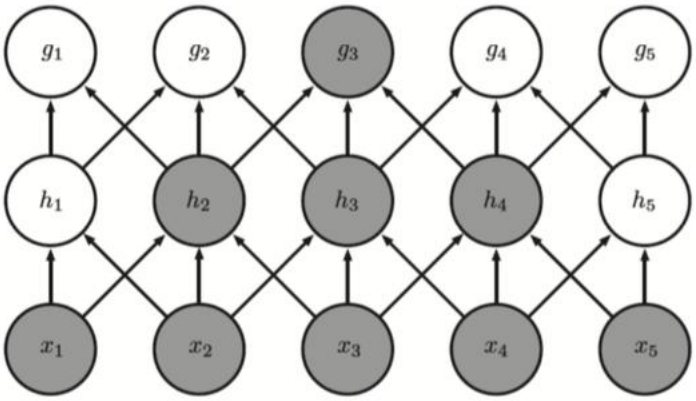


Рисунок 3.5 – опосередкована взаємодія блоків

Розділення параметрів.

В мережі один і той самий параметр може використовуватись в декількох модельних функціях. Наприклад в згортковій мережі кожний елемент ядра використовується для кожного міста входу, отже згортка стає набагато ефективнішою за множенням матриць з точки зору зменшення вимог до пам’яті[16].

В той самий час, в традиційній мережі, кожний елемент використовується одного разу, в час коли множиться з елементом вхідного сигналу, для обчислення вихідного сигналу з шара.

На рисунках 3.6 - 3.7 наведено приклад розділення параметрів.

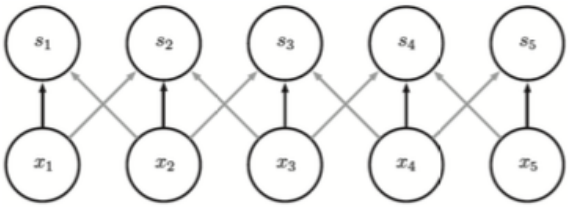


Рисунок 3.6 – один з прикладів використання розділення параметрів

На цьому рисунку ми бачимо приклад використання 3-елементного ядра в згортоковій моделі.

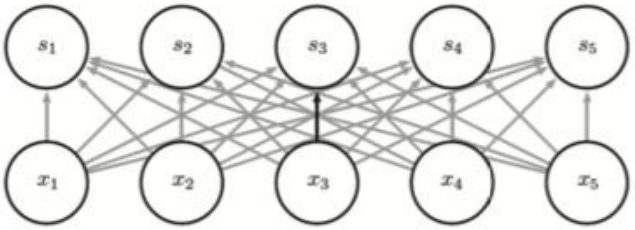


Рисунок 3.7 – приклад відсутності розділення параметрів

Тут навпаки розділення параметрів немає, і параметр використовується один раз.

3.4.2 Операції згортки і пулінгу

Типовий шар згорткової мережі складається з трьох складових (рис. 3.8) [13]. Перша складова – це паралельне виконання кількох згорток, в результаті отримуємо множину лінійних активацій. Друга складова полягає в тому, що кожна лінійна активація пропускається через нелінійну функцію активації, наприклад функцію ReLU лінійної ректифікації. Ця стадія ще називається детекторною [13]. Третя складова – це функція пулінгу для подальшої модифікації виходу шару.

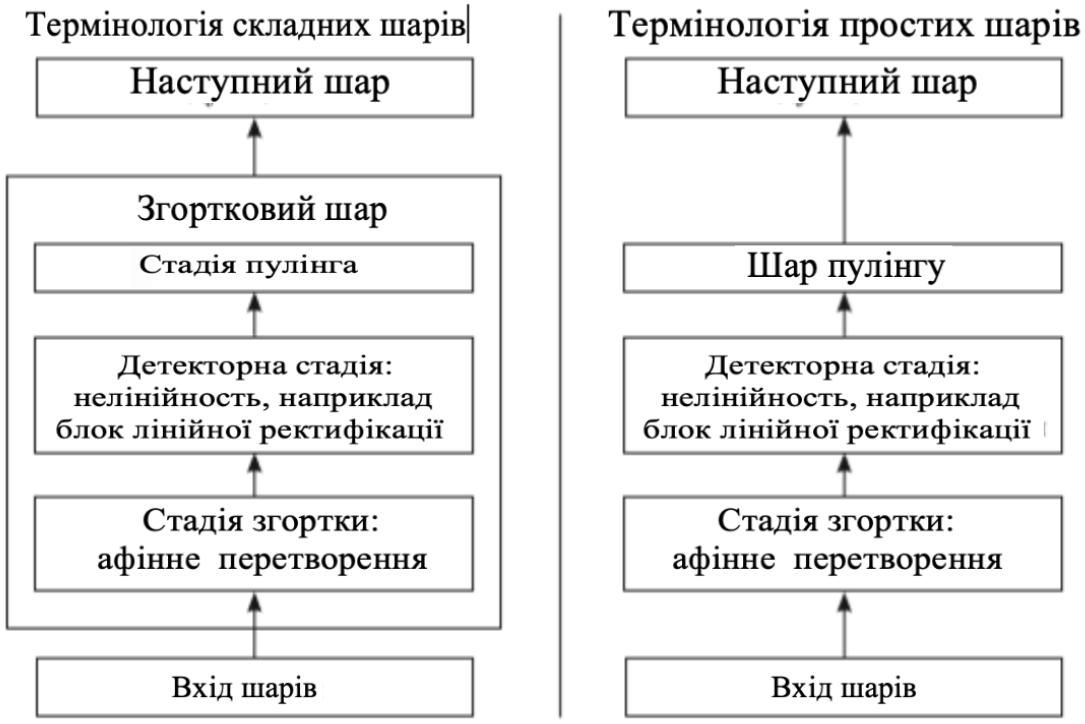


Рисунок 3.8 – Компоненти типового шару згорткової нейронної мережі.

Для опису таких шарів застосовується двояка термінологіях[13]. (Зліва) У цій термінології згорткову мережу розглядається як невеликий набір щодо складних шарів, кожен з яких має багато «стадій». При цьому існує взаємно однозначна відповідність між ядерними тензорами і шарами мережі. У цій книзі ми в основному дотримуємося такої термінології. (Праворуч) У цій термінології згорткову мережу розглядається як великий набір простих верств шарів, а кожен крок обробки вважається повноправним шаром[13]. Це означає, що не у кожного «шару» є параметри.

Функція пулінгу замінює вихід мережі зведеною статистикою прилеглих виходів [22]. Наприклад, операція max-пулінгу повертає максимальне значення в прямокутному околі. Також в якості функцій пулінгу використовуються середнє значення, а також L^2-норма в прямокутному околі і зважене середнє з вагами, які залежать від відстані до центрального елементу.

Мета пулінгу полягає в тому, щоб зробити представлення локально інваріантним щодо малих паралельних переносів вхідного образу. При цьому інваріантність щодо паралельного переносу означає, що якщо змістити вхідне значення на невелику величину, то значення більшості результатів пулінгу не повинно змінитися. На рис. 3.9 надалі показаний приклад роботи цього механізму. Локальна інваріантність, щодо паралельного переносу корисна, якщо більш цікавим є факт існування певної ознаки, а не її місцезнаходження на зображенні. Розглянемо приклад, в якому визначається присутність обличчя на зображенні. В цьому випадку положення очей з точністю до пікселя не є важливим. Потрібно, наприклад, лише знати наявність очей зліва і справа [13].

В інших задачах важливо зберегти місце розташування ознаки. Наприклад, шукається кутова точка, що утворена перетином двох кордонів,

орієнтованих певним чином. Тоді положення кордонів має зберігатися дуже точно, для того щоб можна було перевірити, чи перетинаються вони.

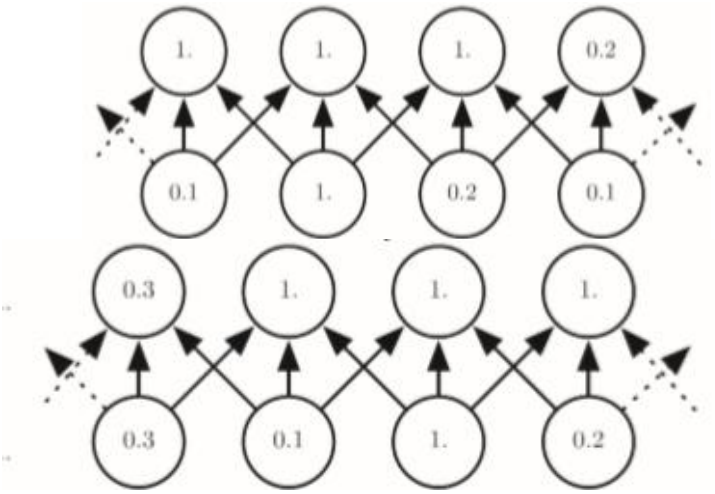


Рисунок 3.9 – Max-пулінг привносить інваріантність.

Стадія пулінга/Детекторна стадія

Max-пулінг привносить інваріантність. Вгорі на (рис. 3.9) знаходиться середина вихідного шару згорткової мережі. У нижньому рядку показані виходи нелінійності, а у верхній – виходи max-пулінг з кроком в один піксель між областями ПУЛІНГ, кожна з яких має ширину три пікселя[15]. Внизу на (рис. 3.8) та ж сама мережа, зрушена вправо на один піксель. У нижньому рядку змінилися всі значення, а у верхній тільки половина, тому що блоки maxпулінга чутливі лише до максимального значення в своїй околиці, а не точному положенню цього значення.

Пулінг можна розглядати як додавання наступного нескінченно сильного апріорного припущення [13]. Воно полягає в тому, що навчена згортковим шаром функція повинна бути інваріантна до малих паралельних переносів. Вважається, що якщо це припущення виконується, то воно може істотно покращити статистичну ефективність мережі [16]. Пулінг за просторовими областями в цілому породжує інваріантність до паралельних перенесень. Однак, якщо такий пулінг проводиться по виходах згорток з

різними параметрами, то ознаки можуть навчитися, до яких перетворень стати інваріантними (рис. 2.9).

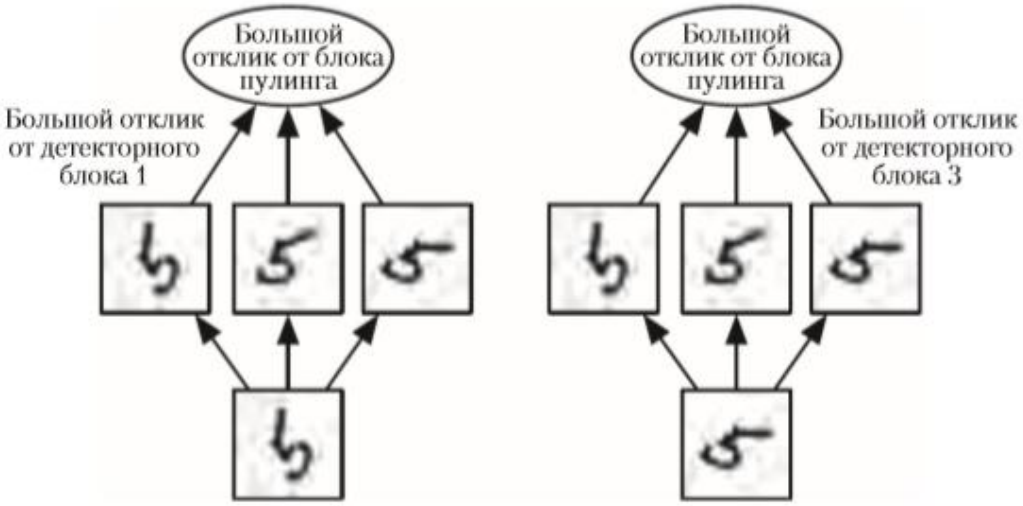


Рисунок 3.10 – Приклад навченої інваріантності

Блок, який виконує пулінг за кількома ознаками, навченим з різними параметрами, може навчитися інваріантності до перетворень входу. Тут ми бачимо набір з трьох навчених фільтрів і блок max-пулінг, який навчився інваріантності до обертання. Всі три фільтра призначені для розпізнавання рукописного цифри 5[15]. Кожен фільтр налаштований на свою орієнтацію п'ятірки. Якщо на вході з'являється цифра 5, то відповідний фільтр розпізнає її, що дасть великий відгук на детекторний блок. Тоді блок max-пулінг дасть великий відгук незалежно від того, який детекторний блок був активований. На малюнку показано, як мережа обробляє два різних входу, що активують різні детекторні блоки. В обох випадках вихід блоку приблизно однаковий. Цей принцип використовується в maxout-мережах та інших згорткових мережах. Махпулінг по просторової області має природну інваріантність до паралельних перенесень. Такий багатоканальний підхід необхідний тільки для навчання іншим перетворенням.

3.5 Спеціалізовані згорткові мережі для отримання ознак зображень

3.5.1 Мережа VGG16. Архітектура мережі. Переваги і обмеження мережі

VGG16 – модель згорткової нейронної мережі, запропонована K. Simonyan і A. Zisserman з Оксфордського університету в статті "Very Deep Convolutional Networks for Large-Scale Image Recognition" (рис.3.11). Модель досягає точності 92.7% – топ-5, при тестуванні на ImageNet в задачі розпізнавання об'єктів на зображенні[14]. Цей датасета є одним з найпопулярніших у світі для тестування нейронних мереж, які вирішують задачу класифікації та складається з більш ніж 14 мільйонів зображень, що належать до 1000 категорій.

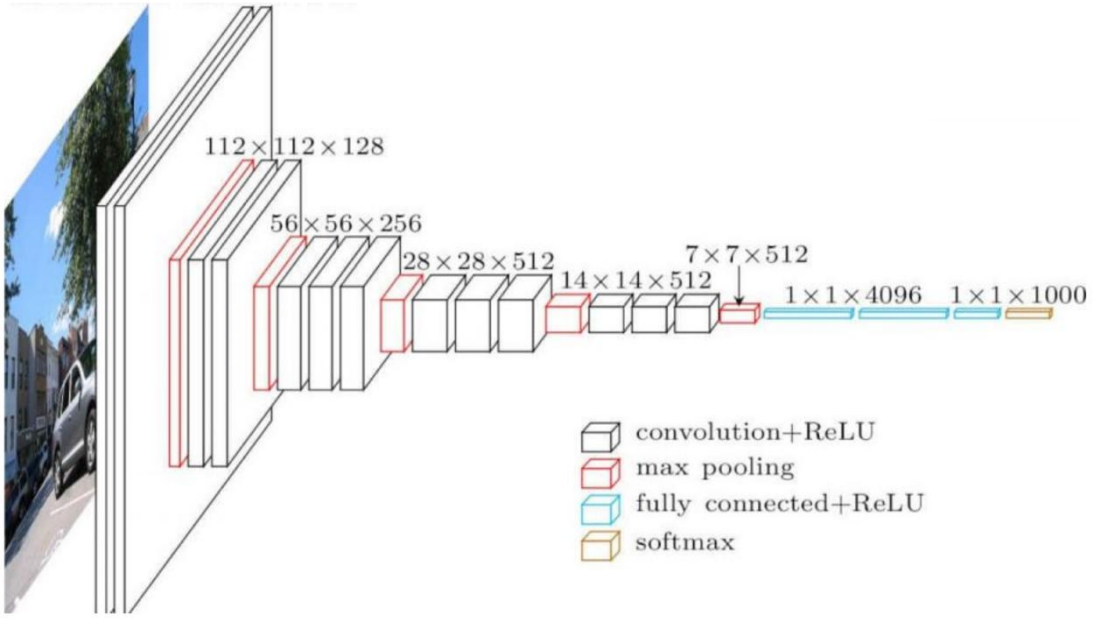


Рисунок 3.11 – Архітектура мережи VGG16

VGG16 – модель згорткової нейронної мережі, запропонована K. Simonyan і A. Zisserman з Оксфордського університету в своїй оригінальній статті "Very Deep Convolutional Networks for Large-Scale Image Recognition". В [14] зазначається, що така модель досягає точності 92.7% – цей результат

входить до п’яти найкращих при тестуванні на навчальному наборі ImageNet в задачі розпізнавання зображень. Цей навчальний набір – один з найпопулярніших у світі для тестування нейронних мереж, які вирішують задачу класифікації.

VGG16 одна з найзнаменітіших моделей нейронной мережі, вона була відправлена на змагання ILSVRC 2014, ця модель є покращеною версією AlexNetб де були замінені великі фільтри (розміру 11x11 та 5x5 у першому та другому згортковому шарі, відповідно), на кілька фільтрів розміром 3x3, наступних один за одним. Для використання на цьому змаганні модель навчалась протягом декількох тижнів завдяки дуже потужним відеокартам компанії NVIDIA.

Архітектура даної моделі подана на рисунку 3.12 має на вході на шар conv1 подаються зображення RGB 224x224. Далі зображення проходять через стек згортальних шарів, в яких використовуються фільтри з дуже маленьким рецептивних полем розміру 3x3 (який є найменшим розміром для отримання уявлення про те, де знаходиться право/ліво, верх/низ, центр).

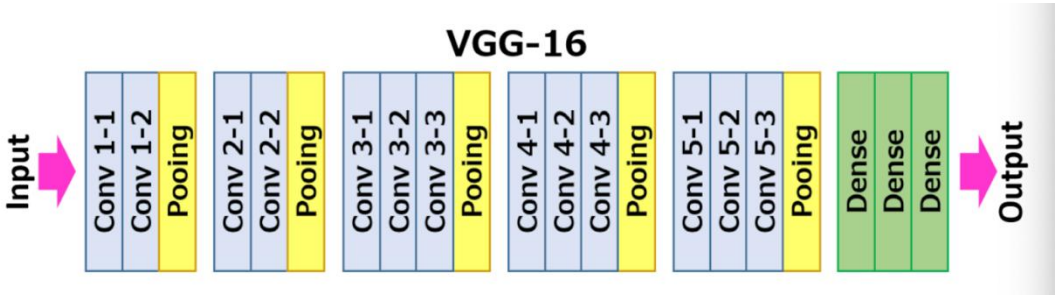


Рисунок 3.12 – Модель

Може використовуватися також згортковий фільтр розміром 1x1, який представляється як лінійне перетворення вхідних каналів з подальшою нелінійносттю. Згортковий крок фіксується на значенні 1 піксель. Для входу згорткового шару вибирається просторове доповнення (padding) так щоб зберегти просторову роздільну здатність після згортки. Тобто, доповнення тепер дорівнює 1 для згорткових шарів розмірності 3x3. Просторовий пулінг здійснюється за допомогою п'яти шарів max-pooling, які слідують за одним із

згорткових шарів, бо не всі згорткові шари мають наступні max-pooling.

Після стеку згортальних шарів (який має різну глибину в різних архітектурах) йдуть три повнозв'язних шару: перші два мають по 4096 каналів, третій – 1000 каналів. Це тому що призначення моделі полягає у класифікації об'єктів за 1000 категоріями; отже, кожному класу відповідає свій один канал. Останнім шаром у таких задачах виступає шар soft-max. Конфігурація повнозв'язних шарів у всіх нейромережах одна і та ж.

Всі приховані шари забезпечені ReLU [16]. Також необхідно відзначити, що мережі (за винятком однієї) не містять шару нормалізації (Local Response Normalisation), так як нормалізація не покращує результату на датасета ILSVRC, а веде до збільшення споживання пам'яті та часу виконання коду.

Конфігурації згортальних мереж представлені на рисунку 3.13. Кожна мережа відповідає своєму імені (AE). Усі зміни мають загальну конструкцію, представлену в архітектурі, і розрізняються лише глибиною: від 11 шарів з вагами в мережі A (8 згортальних і 3 повнозв'язних шару) до 19 (16 згортальних і 3 повнозв'язних шару). Ширина згортальних шарів (кількість каналів) відносно невелика: від 64 у першому шарі до 512 в останньому зі збільшенням кількості каналів в 2 рази після кожного max-pooling шару.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					

Рисунок 3.13 – Конфігурація моделі VGG16

Недоліки мережі:

Дуже повільна швидкість навчання

Сама архітектура мережі важить надто багато (з’являються проблеми пам’яті та пропускної спроможності)

3.5.2 Мережа SqueezeNet

Існує декілька стратегій оптимізації відносно інших моделей[20]:

1. Заміна фільтрів 3×3 на 1×1.

Враховуючи бюджет певної кількості згорткових фільтрів, ми можемо вибрати більшість цих фільтрів 1×1, оскільки фільтр 1×1 має на 9×менше параметрів, ніж фільтр 3×3.

2. Розглянемо шар згортки, який повністю складається з фільтрів розмірності 3×3. Загальна кількість параметрів у цьому шарі дорівнює **добутку** кількості вхідних каналів на кількість фільтрів на добуток 3×3**.** Ми можемо зменшити кількість вхідних каналів до 3×3 фільтрів за допомогою шарів стискання , вони будуть розглянути далі.

Відображати пізній вибір в мережі, для того щоб шари згортки мали великі карти активації.

Ідея полягає в тому, що великі карти активації внаслідок затримки зменшення тиску можуть призвести до підвищення точності класифікації.

Модуль стискання (3.14), який ще називають burn модуль, складається з шару згортки стискання (лише з фільтрами 1×1), який подається в шар розширення, який складається з міксу фільтрів згортки 3×3 та 1×1.В цьому модулі три налагоджуємих гіперпараметра s1×1 відповідає за кількість фільтрів 1×1 в першому шарі, e1×1 і e3×3: кількість 1×1 і 3×3 в розширюваному шарі.

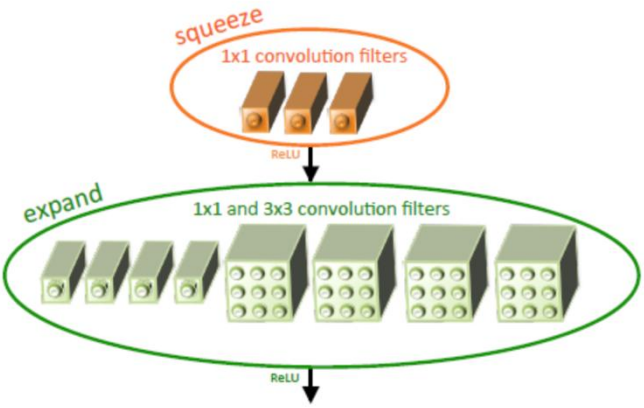


Рисунок 3.14 – модуль стискання з гіперпараметрами: s1×1 = 3, e1×1 =4, e3×3 =5

Під час використання встановлюється s1×1 < (e1×1 + e3×3), тому що шар стискання допомагає обмежити кількість вхідних каналів до фільтрів 3×3, відповідно до 2 стратегії.

На рисунку 3.15 зображені 3 архітектури SqueezeNet.

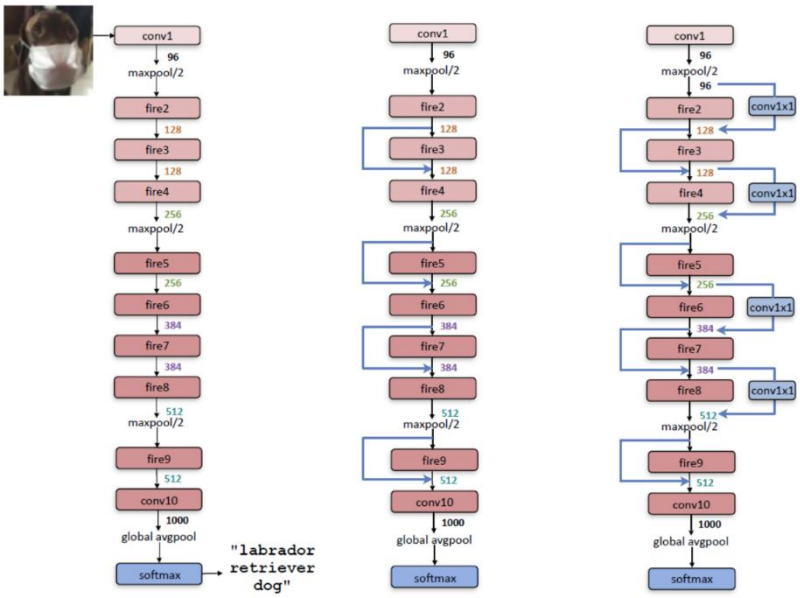


Рисунок 3.15 – Архітектура SqueezeNet (зліва), SqueezeNet з простим обходом(по середені) та SqueezeNet зі складним обходом (праворуч).

SqueezeNet починається з окремого шару згортки (conv1), за ним йде 8 модулів стискання (fire2-9) і закінчується кінцевим шаром згортки (conv10). Кількість фільтрів на цей модуль поступово збільшується від початку до

кінця мережі. Макс-пулінг відбувається з кроком 2 після шарів conv1, fire4, fire8 та conv10.

На рисунку 3.16 наведено статистичну оцінку використання цієї моделі.

CNN architecture	Compression Approach	Data Type	Original → Compressed Model Size	Reduction in Model Size vs. AlexNet	Top-1 ImageNet Accuracy	Top-5 ImageNet Accuracy
AlexNet	None (baseline)	32 bit	240MB	1x	57.2%	80.3%
AlexNet	SVD (Denton et al., 2014)	32 bit	240MB → 48MB	5x	56.0%	79.4%
AlexNet	Network Pruning (Han et al., 2015b)	32 bit	240MB → 27MB	9x	57.2%	80.3%
AlexNet	Deep Compression (Han et al., 2015a)	5-8 bit	240MB → 6.9MB	35x	57.2%	80.3%
SqueezeNet (ours)	None	32 bit	4.8MB	**50x**	57.5%	80.3%
SqueezeNet (ours)	Deep Compression	8 bit	4.8MB → 0.66MB	**363x**	57.5%	80.3%
SqueezeNet (ours)	Deep Compression	6 bit	4.8MB → 0.47MB	**510x**	57.5%	80.3%

Рисунок 3.16 – Статистичні показники моделі порівняно с іншими моделями

За допомогою SqueezeNet досягається зменшення розміру моделі більш ніж у 50 разів відносно розміру AlexNet, при цьому дотримаючись та перевищуючі показники точності.

2.5.3 Мережа ResNet

Робота архітектури ResNet виділяється наступним чином:

- на вихід подаються 2 згорткових шари;
- обхід 2-х згорткових шарів.

Особливість роботи зображена на рисунку 3.17.

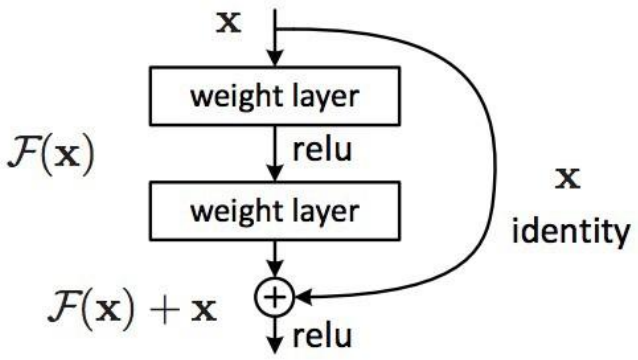


Рисунок 3.17 – Особливість роботи ResNet

Такий підхід не виглядає як щось нове, але головне це обхід двух шарів і можливість застосування у разі наявності великого об'єму даних. І саме обхід двох щарів є головним ключем поліпшення, тому що обхід одного шара не дав особливих поліпшень. На 2-х шарах можна розглянути невеликий класифікатор або Network in network! [21]

ResNet з великою кількістю шарів почав використовувати шар схожий на «вузьке місце» (рис. 3.18).

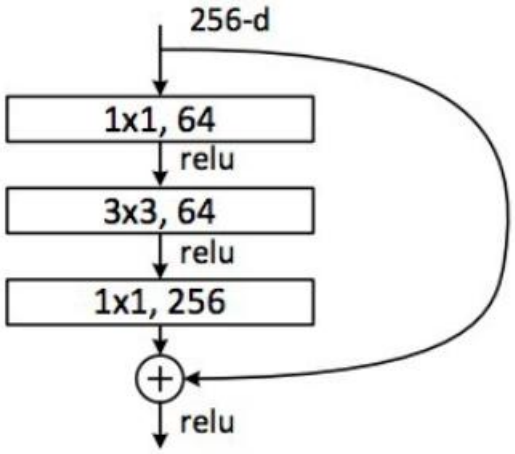


Рисунок 3.18 – Шар «вузького місця» для ResNet

Завдяки цьому шару зменшується кількість ознак всіх шарів, спочатку використовується згортка 1x1 з виходом у розмірі ¼ від класичного а потім 3x3 шаром, а потім знову згортання 1x1 на більшу кількість функцій. Завдяки цьому обчислення залишаються малими, зберігаючи при цьому багате поєднання функцій.

Існують різні варіації кількості шарів мережи (рис. 3.19). Кожен блок мережи має два рівня глибини (для невеликих мереж як ResNet 18,34) або 3 рівня (ResNet 50, 101, 152).

50-шарова ResNet : кожен 3-шаровий блок замінюється в 34-шарової мережі цим 3-шаровим вузьким місцем, в результаті виходить 50-шарова ResNet. Ця модель має 3,8 мільярда FLOPs.[22]

ResNet з 101 і 152 шарами: моделі створюються, використовуючи більше 3-шарових блоків. Навіть після збільшення глибини 152-шарова ResNet (11,3 мільярда FLOP) має меншу складність, ніж мережі VGG-16/19 (15,3/19,6 мільярда FLOPs).

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Рисунок 3.19 – варіації кількості шарів моделі ResNet

3.6 Висновки

В розділі наведено математичне забезпечення глибоких нейронних мереж прямого розповсюдження. Розглянуто побудову скритих шарів ReLU і вихідного шару softmax мережі. Вивчено алгоритми оптимізації глибоких нейронних мереж, такі як стохастичний градієнтний спуск та метод Adam, а також сучасні методи Глоро і Хе для ініціалізації ваг. Описано основи згорткових нейронних мереж, операції згортки і пулінгу.

РОЗДІЛ 4 АРХІТЕКТУРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ РОБОТИ

4.1 Обґрунтування вибору бібліотеки

На рисунку 4.1 наведено усі підключені для реалізації програмного продукту бібліотеки

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import VGG16
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.layers import AveragePooling2D
from tensorflow.keras.layers import Dropout
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Input
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.utils import to_categorical
from sklearn.preprocessing import LabelBinarizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from imutils import paths
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import random
import shutil
import cv2
import os
```

Рисунок 4.1 – список підключених бібліотек

TensorFlow – це відкрита платформа з відкритим кодом для машинного навчання. Ця бібліотека має вичерпну гнучку екосистему інструментів, бібліотек та ресурсів громади, що дозволяє дослідникам відкривати нові можливості в ML (Машинне навчання) та розробникам легко створювати та розгортати програми, що працюють на ML.

Tensorflow був обраний, тому що в ньому є реалізовний API Keras, також завдяки цієї лібі можно легко реалізовувати власні моделі. Архітектура платформи дуже гнучка та проста для того щоб переносити свої ідея та концепції до коду.

Sklearn – в випадку цього дослідження, ця бібліотека грає роль допоміжною в створенні репортів якости навчання та використовується для предпроцесінгу даних.

Pandas – дуже гнучка бібліотека для аналізу, маніпуляцій та роботи за набором даних

NumPy – библиотека з широкими можливостями в області научно-технічних досліджень

Matplotlib – використовується для реалізації графіків

4.2 Опис критеріїв якості класифікації

Попередні позначення:

*TP - істинно позитивно;

*TN - істинно негативно;

*FP - помилково позитивно;

*FN - помилково негативно.

Розглянемо метрики які використовуються прі оцінці якості класифікації. В нашому випадку вони такі:

Точність (англ. accuracy) – показник точності роботи моделі на тренувальної та валідаційній виборці

$$\text{accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \quad (4.1)$$

Точність один до одного (англ. Precision) – показник точності, але на відмін від accuracy, показує точність як близькість вимірювань один до одного:

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad (4.2)$$

Чутливість (англ. recall) – частка від загальної кількості відповідних екземплярів які були фактично знайдені:

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (4.3)$$

F1 міра – комбінована міра точності:

$$\text{F1} = \frac{2 * \text{precision} * \text{recall}}{\text{presicion} + \text{recall}} \quad (4.4)$$

4.3 Побудова моделей згорткової нейронної мережі для класифікації рентгенів хворих на COVID

4.3.1 Використання дропауту та нормалізації

На Рисунку 4.1 на далі наведено реалізацію використання дропауту, активації та нормалізації.

```
bnmomemtum=0.9
y = tf.keras.layers.BatchNormalization(momentum=bnmomemtum)(y)

      ....


y = tf.keras.layers.Dropout(0.5)(y)
y = tf.keras.layers.Dense(2, activation='softmax')(y)
```

Рисунок 4.1 – реалізація дропауту, активації та нормалізації

У якості експерименту було побудовано однакові моделі мережи SqueezeNet, але одна з використанням Dropout, а інша – без. В результаті чого трапилось перенавчання, далі на рисунку 4.2 – 4.3, наведено порівняння якості навчання у двох випадках.

	loss	acc	val_loss	val_acc
EPOCH 1	0.452237	0.785	1.437321	0.50
EPOCH 2	0.413548	0.850	0.677107	0.64
EPOCH 3	0.420252	0.835	0.363857	0.84
EPOCH 4	0.348809	0.865	0.369369	0.80
EPOCH 5	0.319194	0.880	1.517003	0.52
EPOCH 6	0.315752	0.885	0.487907	0.76
EPOCH 7	0.245385	0.920	0.275911	0.86
EPOCH 8	0.231226	0.915	1.497127	0.66
EPOCH 9	0.202359	0.920	2.053224	0.50
EPOCH 10	0.282645	0.885	0.530321	0.66

Рисунок 4.2 – показники точності та втрат при навчанні без дропаута

	loss	acc	val_loss	val_acc
EPOCH 1	0.405569	0.865	2.149067	0.50
EPOCH 2	0.343202	0.875	1.606079	0.54
EPOCH 3	0.296488	0.870	1.634704	0.56
EPOCH 4	0.326626	0.890	0.947183	0.56
EPOCH 5	0.296084	0.885	0.339898	0.84
EPOCH 6	0.255191	0.910	1.274313	0.52
EPOCH 7	0.259797	0.915	2.215011	0.54
EPOCH 8	0.186517	0.930	0.220697	0.90
EPOCH 9	0.244209	0.920	1.226965	0.68
EPOCH 10	0.246801	0.900	1.755334	0.56

Рисунок 4.3 – показники точності та втрат при навчанні з дропаутом

4.3.2 Побудова моделей CNN

На рисунку 4.4 реалізований приклад побудови та компіляції моделі VGG16 (частично реалізованої в стандартній бібліотеці).

```
baseModel = VGG16(weights="imagenet", include_top=False, input_tensor=Input(shape=(224, 224, 3)))

headModel = baseModel.output
headModel = AveragePooling2D(pool_size=(4, 4))(headModel)
headModel = Flatten(name="flatten")(headModel)
headModel = Dense(64, activation="relu")(headModel)
headModel = Dropout(0.5)(headModel)
headModel = Dense(2, activation="softmax")(headModel)

model = Model(inputs=baseModel.input, outputs=headModel)

for layer in baseModel.layers:
    layer.trainable = False
```

```
opt = Adam(lr=INIT_LR, decay=INIT_LR / EPOCHS)
model.compile(loss="binary_crossentropy", optimizer=opt, metrics=["accuracy"])
```

Рисунок 4.4 – приклад побудови та компіляції моделі VGG16

Модель SqueezeNet не має частичної або повної реалізації в бібліотеках, тому була реалізована власноруч, надалі на рисунку 4.5 наведено реалізацію моделі

```
def fire(x, squeeze, expand):
  y  = tf.keras.layers.Conv2D(filters=squeeze, kernel_size=1, activation='relu', padding='same')(x)
  y = tf.keras.layers.BatchNormalization(momentum=bnmomemtum)(y)
  y1 = tf.keras.layers.Conv2D(filters=expand//2, kernel_size=1, activation='relu', padding='same')(y)
  y1 = tf.keras.layers.BatchNormalization(momentum=bnmomemtum)(y1)
  y3 = tf.keras.layers.Conv2D(filters=expand//2, kernel_size=3, activation='relu', padding='same')(y)
  y3 = tf.keras.layers.BatchNormalization(momentum=bnmomemtum)(y3)
  return tf.keras.layers.concatenate([y1, y3])

def fire_module(squeeze, ex
  return lambda x:

x = tf.keras.layers.                    # input is 192x192 pixels RGB

y = tf.keras.layers.Conv2D(kernel_size=3, filters=32, padding='same', use_bias=True, activation='relu')(x)
y = tf.keras.layers.BatchNormalization(momentum=bnmomemtum)(y)
y = fire_module(24, 48)(y)
y = tf.keras.layers.MaxPooling2D(pool_size=2)(y)
y = fire_module(48, 96)(y)
y = tf.keras.layers.MaxPooling2D(pool_size=2)(y)
y = fire_module(64, 128)(y)
y = tf.keras.layers.MaxPooling2D(pool_size=2)(y)
y = fire_module(48, 96)(y)
y = tf.keras.layers.MaxPooling2D(pool_size=2)(y)
y = fire_module(24, 48)(y)
y = tf.keras.layers.GlobalAveragePooling2D()(y)
y = tf.keras.layers.Dropout(0.5)(y)
y = tf.keras.layers.Dense(2, activation='softmax')(y)

model = tf.keras.Model(x, y)

model.compile(
  optimizer='adam',
  loss= 'binary_crossentropy',
  metrics=['accuracy'])
```

Рисунок 4.5 – реалізація моделі SqueezeNet

Спочатку побудовано функція яка генерує той самий модуль стискання, і в наслідку реалізована модель та її компіляція.

На вході реалізовано вступний шар, де подається зображення 224x224 пікселя.

Далі йде згортковий шар Conv2D, з розміром кернела = 8 та використанням функції активацій ReLu. Після чого відтворен шар нормалізації по батчам. За цим йде набір модулів стискання (англ. burn module) на пару из шарами макс-пулінгу.

Передостанній шар в реалізації – це дропаут, експеримент з використання якого описано в попередньому підрозділі.

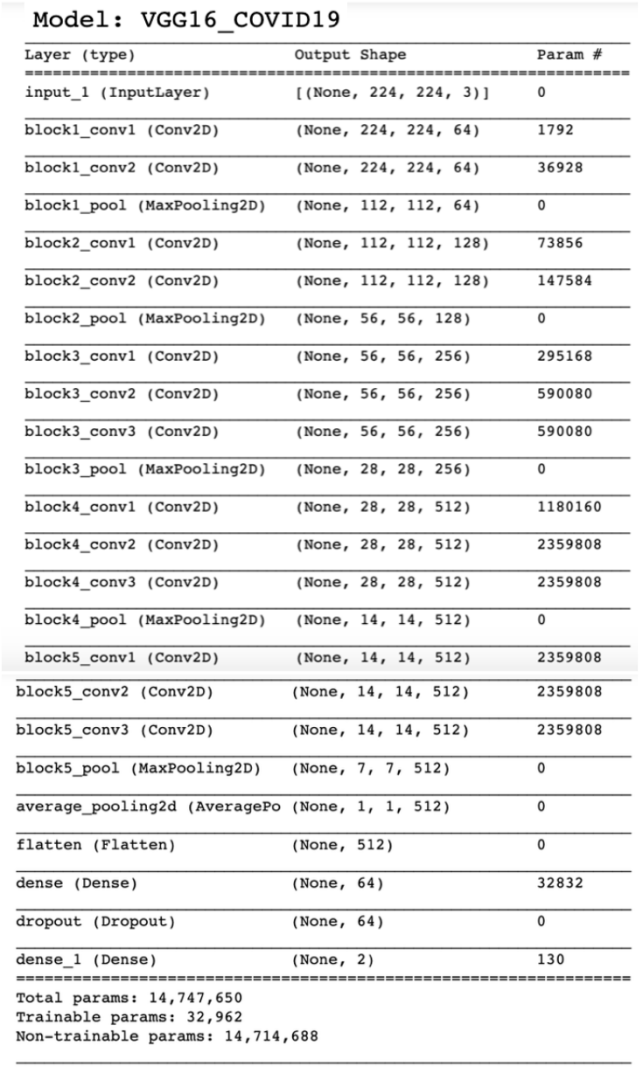
Що стосовно компіляції, в ній використовується метод Адам в якості оптимізатора, що дає кращий відсоток точності предикту та навчання.

Функція втрат – бінарна перехресна ентропія, через те що ми класифікуємо зображення на два класи.

4.3.3 Огляд архітектури оптимально побудованих моделей

В цьому підрозділі наведено кінцеві по-шарові вигляди побудованих моделей.

На рисунку 4.6 можна побачити архітектуру моделі VGG16.



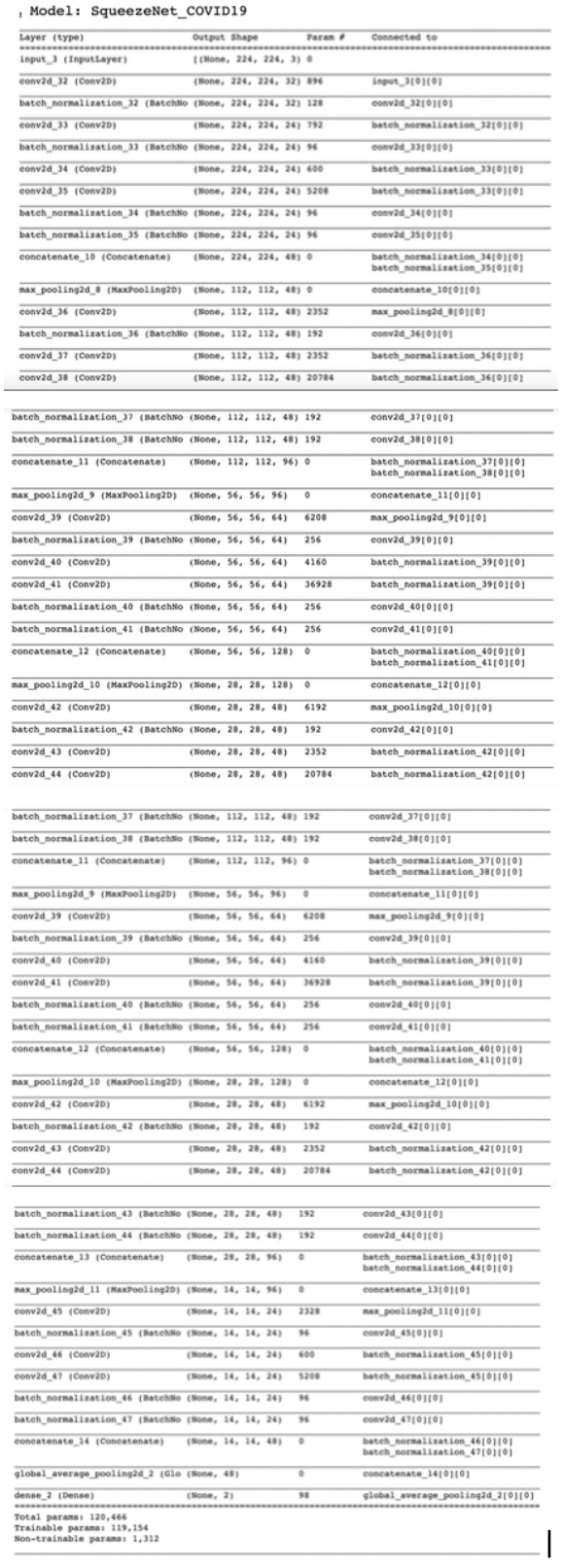
Model: VGG16_COVID19

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 224, 224, 3)]	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1180160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2359808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2359808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2359808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0
average_pooling2d (AveragePo	(None, 1, 1, 512)	0
flatten (Flatten)	(None, 512)	0
dense (Dense)	(None, 64)	32832
dropout (Dropout)	(None, 64)	0
dense_1 (Dense)	(None, 2)	130

Total params: 14,747,650
Trainable params: 32,962
Non-trainable params: 14,714,688

Рисунок 4.6 – архітектура реалізованої VGG16 моделі

На Рисунку 4.7 зображені моделі SqueezeNet.



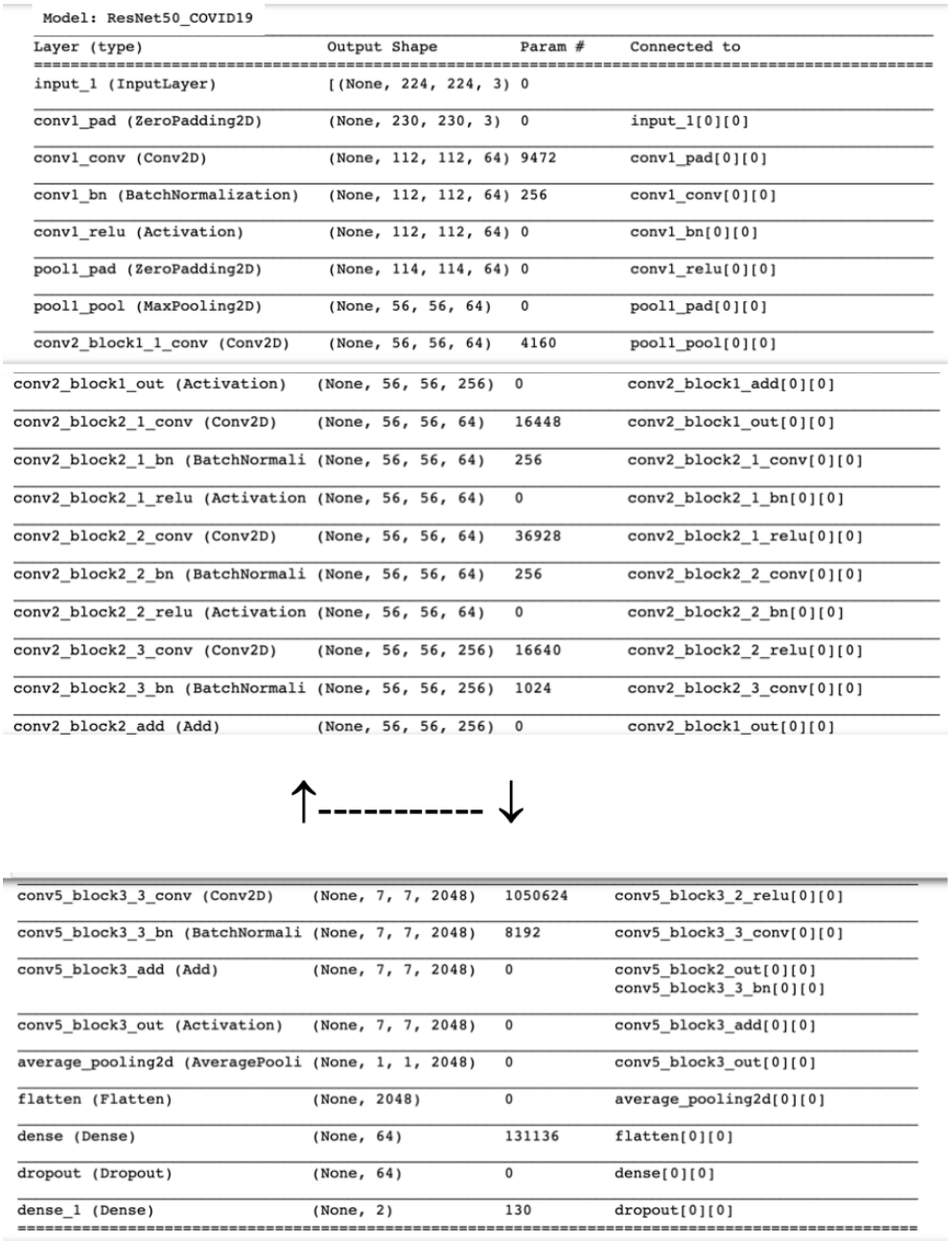
Model: SqueezeNet_COVID19

Layer (type)	Output Shape	Param #	Connected to
input_3 (InputLayer)	[(None, 224, 224, 3)	0	
conv2d_32 (Conv2D)	(None, 224, 224, 32)	896	input_3[0][0]
batch_normalization_32 (BatchNo	(None, 224, 224, 32)	128	conv2d_32[0][0]
conv2d_33 (Conv2D)	(None, 224, 224, 24)	792	batch_normalization_32[0][0]
batch_normalization_33 (BatchNo	(None, 224, 224, 24)	96	conv2d_33[0][0]
conv2d_34 (Conv2D)	(None, 224, 224, 24)	600	batch_normalization_33[0][0]
conv2d_35 (Conv2D)	(None, 224, 224, 24)	5208	batch_normalization_33[0][0]
batch_normalization_34 (BatchNo	(None, 224, 224, 24)	96	conv2d_34[0][0]
batch_normalization_35 (BatchNo	(None, 224, 224, 24)	96	conv2d_35[0][0]
concatenate_10 (Concatenate)	(None, 224, 224, 48)	0	batch_normalization_34[0][0] batch_normalization_35[0][0]
max_pooling2d_8 (MaxPooling2D)	(None, 112, 112, 48)	0	concatenate_10[0][0]
conv2d_36 (Conv2D)	(None, 112, 112, 48)	2352	max_pooling2d_8[0][0]
batch_normalization_36 (BatchNo	(None, 112, 112, 48)	192	conv2d_36[0][0]
conv2d_37 (Conv2D)	(None, 112, 112, 48)	2352	batch_normalization_36[0][0]
conv2d_38 (Conv2D)	(None, 112, 112, 48)	20784	batch_normalization_36[0][0]
batch_normalization_37 (BatchNo	(None, 112, 112, 48)	192	conv2d_37[0][0]
batch_normalization_38 (BatchNo	(None, 112, 112, 48)	192	conv2d_38[0][0]
concatenate_11 (Concatenate)	(None, 112, 112, 96)	0	batch_normalization_37[0][0] batch_normalization_38[0][0]
max_pooling2d_9 (MaxPooling2D)	(None, 56, 56, 96)	0	concatenate_11[0][0]
conv2d_39 (Conv2D)	(None, 56, 56, 64)	6208	max_pooling2d_9[0][0]
batch_normalization_39 (BatchNo	(None, 56, 56, 64)	256	conv2d_39[0][0]
conv2d_40 (Conv2D)	(None, 56, 56, 64)	4160	batch_normalization_39[0][0]
conv2d_41 (Conv2D)	(None, 56, 56, 64)	36928	batch_normalization_39[0][0]
batch_normalization_40 (BatchNo	(None, 56, 56, 64)	256	conv2d_40[0][0]
batch_normalization_41 (BatchNo	(None, 56, 56, 64)	256	conv2d_41[0][0]
concatenate_12 (Concatenate)	(None, 56, 56, 128)	0	batch_normalization_40[0][0] batch_normalization_41[0][0]
max_pooling2d_10 (MaxPooling2D)	(None, 28, 28, 128)	0	concatenate_12[0][0]
conv2d_42 (Conv2D)	(None, 28, 28, 48)	6192	max_pooling2d_10[0][0]
batch_normalization_42 (BatchNo	(None, 28, 28, 48)	192	conv2d_42[0][0]
conv2d_43 (Conv2D)	(None, 28, 28, 48)	2352	batch_normalization_42[0][0]
conv2d_44 (Conv2D)	(None, 28, 28, 48)	20784	batch_normalization_42[0][0]
batch_normalization_43 (BatchNo	(None, 28, 28, 48)	192	conv2d_43[0][0]
batch_normalization_44 (BatchNo	(None, 28, 28, 48)	192	conv2d_44[0][0]
concatenate_13 (Concatenate)	(None, 28, 28, 96)	0	batch_normalization_43[0][0] batch_normalization_44[0][0]
max_pooling2d_11 (MaxPooling2D)	(None, 14, 14, 96)	0	concatenate_13[0][0]
conv2d_45 (Conv2D)	(None, 14, 14, 24)	2328	max_pooling2d_11[0][0]
batch_normalization_45 (BatchNo	(None, 14, 14, 24)	96	conv2d_45[0][0]
conv2d_46 (Conv2D)	(None, 14, 14, 24)	600	batch_normalization_45[0][0]
conv2d_47 (Conv2D)	(None, 14, 14, 24)	5208	batch_normalization_45[0][0]
batch_normalization_46 (BatchNo	(None, 14, 14, 24)	96	conv2d_46[0][0]
batch_normalization_47 (BatchNo	(None, 14, 14, 24)	96	conv2d_47[0][0]
concatenate_14 (Concatenate)	(None, 14, 14, 48)	0	batch_normalization_46[0][0] batch_normalization_47[0][0]
global_average_pooling2d_2 (Glo	(None, 48)	0	concatenate_14[0][0]
dense_2 (Dense)	(None, 2)	98	global_average_pooling2d_2[0][0]

Total params: 120,466
Trainable params: 119,154
Non-trainable params: 1,312

Рисунок 4.7 – архітектура реалізованої SqueezeNet моделі

На Рисунку 4.8 зображено кінцеву та початкову частину архітектури моделі ResNet50, модель наведена не повністю через те, що блоки мають однакові типи та їх дуже багато.



Model: ResNet50_COVID19

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 224, 224, 3)	0	
conv1_pad (ZeroPadding2D)	(None, 230, 230, 3)	0	input_1[0][0]
conv1_conv (Conv2D)	(None, 112, 112, 64)	9472	conv1_pad[0][0]
conv1_bn (BatchNormalization)	(None, 112, 112, 64)	256	conv1_conv[0][0]
conv1_relu (Activation)	(None, 112, 112, 64)	0	conv1_bn[0][0]
pool1_pad (ZeroPadding2D)	(None, 114, 114, 64)	0	conv1_relu[0][0]
pool1_pool (MaxPooling2D)	(None, 56, 56, 64)	0	pool1_pad[0][0]
conv2_block1_1_conv (Conv2D)	(None, 56, 56, 64)	4160	pool1_pool[0][0]
conv2_block1_out (Activation)	(None, 56, 56, 256)	0	conv2_block1_add[0][0]
conv2_block2_1_conv (Conv2D)	(None, 56, 56, 64)	16448	conv2_block1_out[0][0]
conv2_block2_1_bn (BatchNormali	(None, 56, 56, 64)	256	conv2_block2_1_conv[0][0]
conv2_block2_1_relu (Activation	(None, 56, 56, 64)	0	conv2_block2_1_bn[0][0]
conv2_block2_2_conv (Conv2D)	(None, 56, 56, 64)	36928	conv2_block2_1_relu[0][0]
conv2_block2_2_bn (BatchNormali	(None, 56, 56, 64)	256	conv2_block2_2_conv[0][0]
conv2_block2_2_relu (Activation	(None, 56, 56, 64)	0	conv2_block2_2_bn[0][0]
conv2_block2_3_conv (Conv2D)	(None, 56, 56, 256)	16640	conv2_block2_2_relu[0][0]
conv2_block2_3_bn (BatchNormali	(None, 56, 56, 256)	1024	conv2_block2_3_conv[0][0]
conv2_block2_add (Add)	(None, 56, 56, 256)	0	conv2_block1_out[0][0]

↑---------- ↓

conv5_block3_3_conv (Conv2D)	(None, 7, 7, 2048)	1050624	conv5_block3_2_relu[0][0]
conv5_block3_3_bn (BatchNormali	(None, 7, 7, 2048)	8192	conv5_block3_3_conv[0][0]
conv5_block3_add (Add)	(None, 7, 7, 2048)	0	conv5_block2_out[0][0] conv5_block3_3_bn[0][0]
conv5_block3_out (Activation)	(None, 7, 7, 2048)	0	conv5_block3_add[0][0]
average_pooling2d (AveragePooli	(None, 1, 1, 2048)	0	conv5_block3_out[0][0]
flatten (Flatten)	(None, 2048)	0	average_pooling2d[0][0]
dense (Dense)	(None, 64)	131136	flatten[0][0]
dropout (Dropout)	(None, 64)	0	dense[0][0]
dense_1 (Dense)	(None, 2)	130	dropout[0][0]

Рисунок 4.8 – архітектура реалізованої SqueezeNet моделі

4.3.4 Побудова прогнозу захворювання на COVID та результат тестування програмного продукту на зображеннях

На основі кращої моделі, була перевірена тестова вибірка. Результати перевірки рентген зображень цілим скопом на основі цієї моделі наведено на рисунку 4.9.

Матриця
помилок

```
[[25  0]
 [ 3 23]]
acc: 0.9412
sensitivity: 1.0000
specificity: 0.8846
```

Рисунок 4.9 – матриця помилок прогнозування кращої моделі

Виходячи з цієї матриці, можна зробити висновок, що серед 25 рентген зображень здорової людини, усі були класифіковані правильно, в той час як серед 26 зображень вибірки хворих, лише 23 були класифіковані безпомилково.

Загальна точність моделі дорівнює 94,12 %, це можна інтерпретувати як точна класифікація 94 випадків із 100.

Також реалізована перевірка якості прогнозування моделі, при надані рентген-зображення на вхід, та його повернення із результатом прогнозу. Приклад цієї можливості наведено на рисунку 4.10.

Рисунок 4.10 – реалізація можливості передачі продукту рентген-фото та отримання предикту що стосовно наявності захворювання

В цьому випадку обрана випадкова фотографія із датасету рентген-знимків хворих на коронавірус людей, далі виконується предикт на основі кращої моделі.

У випадку, якщо найбільший аргумент поверненого результату дорівнює одиниці, це зображення класифікується як Covid-19.

4.4 Результати класифікації рентгенів хворих на COVID на основі різних архітектур нейронних мереж

4.4.1 Мережа VGG16

Результати прогнозування на основі найкращої моделі SqueezeNet наведено на рисунку 4.11.

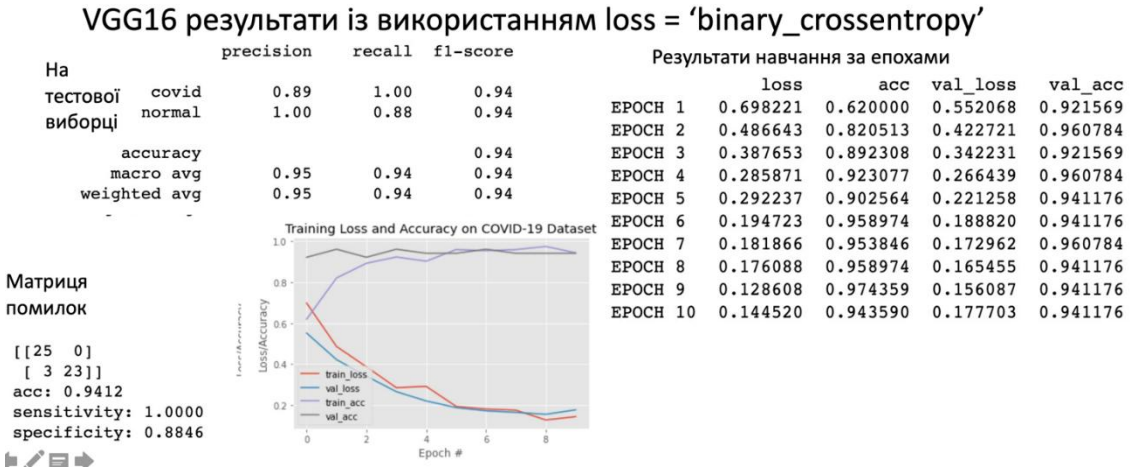


Рисунок 4.11 – Результати навчання та предикту найкращої моделі VGG16

4.4.2 Мережа SqueezeNet

Результати прогнозування на основі найкращої моделі SqueezeNet наведено на рисунку 4.12.

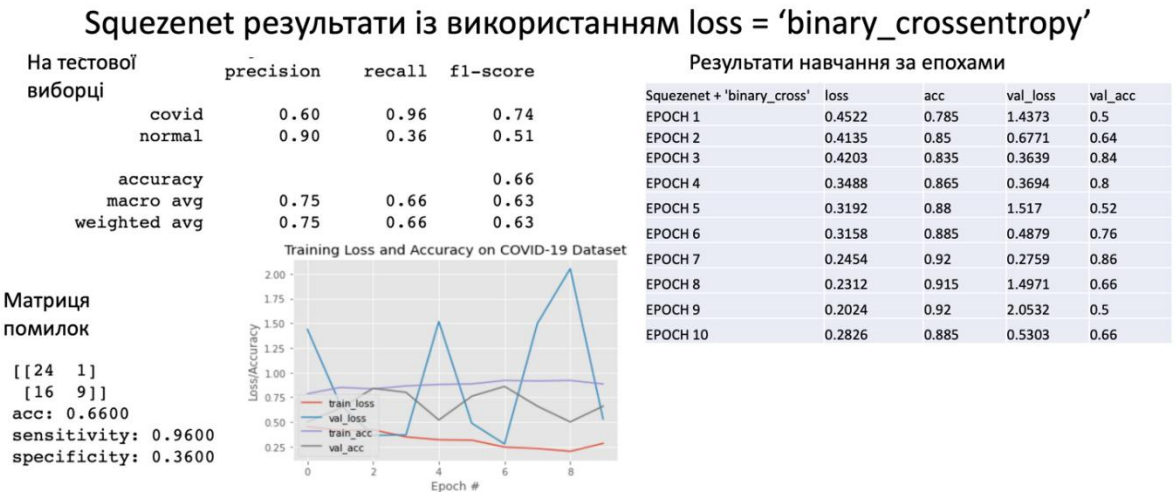


Рисунок 4.12 – Результати навчання та предикту найкращої моделі SqueezeNet

4.4.3 Мережа ResNet50

Результати прогнозування на основі найкращої моделі SqueezeNet наведено на рисунку 4.13.

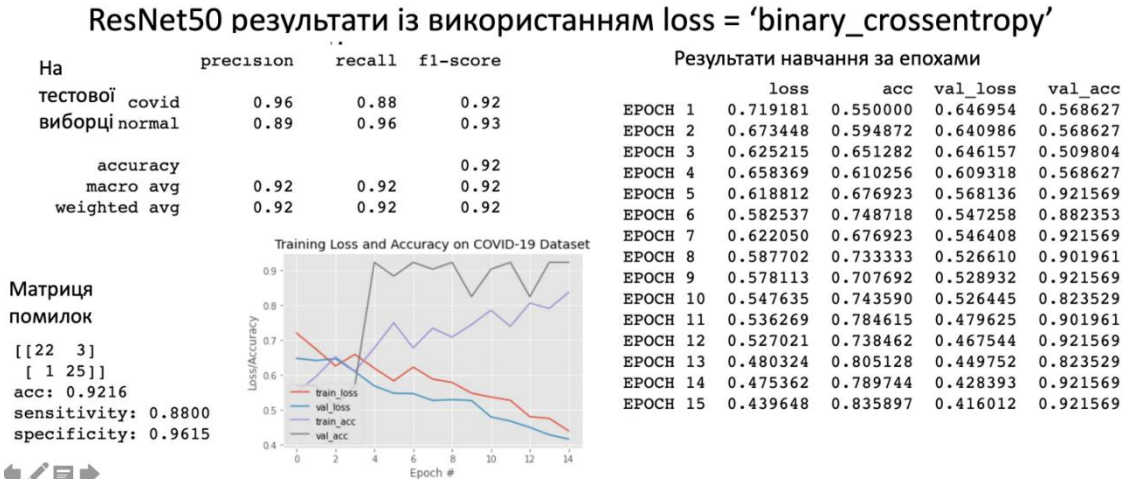


Рисунок 4.13 – Результати навчання та предикту найкращої моделі ResNet50

4.5 Висновки

В даному розділі оглянуто процес реалізація програмного продукту, на основі теорії яка була розглянута у другому розділі. Розглянуто набір бібліотек які використовуються та показників якості класифікації. Отримані результати мають різні значення якості, що демонструє дійсно різний підхід до навчання моделей. Результати отримані за найкращою моделлю дуже гарні, отримано 94% якості класифікації. На основі цього, можна зробити висновок що програмний продукт реалізовано доцільно та він є закінченим, та те, що використання згорткових нейронних мереж має місце бути при розпізнаванні рентген зображень хворих на COVID-19.

РОЗДІЛ 5 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

Вступ

Даний розділ присвячений функціонально-вартісному аналізу програмного продукту, призначеного для розпізнавання чинників захворювання на коронавірусну інфекцію COVID-19 за рентген знімком. Програмний продукт був розроблений за допомогою мови програмування Python, в якості IDE був обраний Jupyter Notebook.

Програмний продукт призначено для використання на персональних комп'ютерах під управлінням операційної системи Windows, MacOS або Linux.

5.1 Постановка задачі техніко-економічного аналізу

У роботі застосовується метод ФВА для проведення техніко-економічного аналізу розробки моделі розпізнавання чинників захворювання на коронавірусну інфекцію COVID-19. Оскільки основні проектні рішення стосуються всієї системи, кожна окрема підсистема має їм задоволняти.

Під час застосування методу ФВА для проведення технікоекономічного аналізу розробленого продукту необхідно обрати систему показників якості програмного продукту. Він має відповідати наступним технічним вимогам:

- функціонування на персональних комп'ютерах зі стандартним набором компонент;
- швидкодія та обробка великого об'єму даних у режимі реального часу;
- зручність та простота у роботі для користувача або для розробника іншого програмного забезпечення на основі програмного продукту, що аналізується;
- мінімальні витрати на впровадження програмного продукту.

5.2 Обґрунтування функцій та параметрів програмного продукту

Виходячи з конкретних цілей, які реалізуються:

F1 – вибір мови програмування: а) Python, б) C#.

F2 – використання готових бібліотек: а) використання готової бібліотеки,

б) розробка алгоритмів для створення нейронних мереж вручну

F3 – середовище розробки: а) PyCharm, б) Jupyter.

Виходячи з представлених варіантів будуємо морфологічну карту (рис.5.1).

Виходячи з конкретних цілей, які реалізуються :

F1 – вибір мови програмування: а) Python, б) C#.

F2 – використання готових бібліотек: а) використання готової бібліотеки,

б) розробка алгоритмів для створення нейронних мереж вручну

F3 – середовище розробки: а) PyCharm, б) Jupyter.

Виходячи з представлених варіантів будуємо морфологічну карту (рис.5.1).

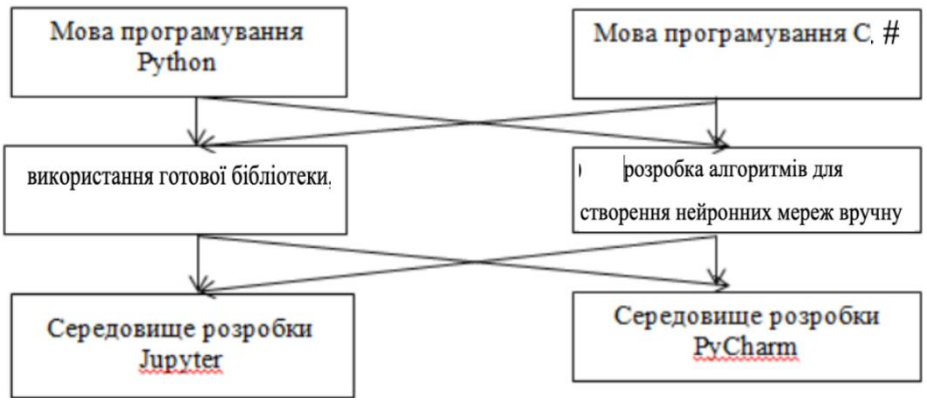


Рисунок 5.1 – Морфологічна карта

Спираючись на карту була побудована позитивно-негативна матриця (табл. 5.1).

Таблиця 5.1 – Позитивно-негативна матриця

Основна функція	**Варіант реалізації**	**Переваги**	**Недоліки**

F1	А	Більше пристосований для роботи з великими даними, кросплатформний	Динамічна типізація
	Б	Відсутня динамічна типізація	Більше часу для написання коду, погано працюю з великими даними
F2	А	Легкість реалізації, економія часу	
	Б	Оптимально для власних продуктів	Велика кількість помилок, затрачений час
F3	А	Широкий вибір можливостей	Відсутня відладка коду
	Б	Відлатка коду	Додаткова інсталяція

На основі аналізу позитивно-негативної матриці робимо висновок, що при розробці програмного продукту деякі варіанти реалізації функцій варто відкинути, тому, що вони не відповідають поставленим перед програмним продуктом задачам. Ці варіанти відзначені у морфологічній карті.

Функція F1:

Оскільки розрахунки проводяться з великими об’ємами вхідних даних, то час виконання програмного коду є дуже необхідним, тому варіант Б має бути відкинутий.

Функція F2:

Оскільки реалізація існуючих бібліотек не визиває сумніву в їх якості, то варіант б має бути відкинутий

Функція F3:

Інтерфейс користувача не відіграє велику роль у даному програмному продукту, тому вважаємо варіанти А та Б гідними розгляду.

Таким чином, будемо розглядати такі варіанти реалізації ПП:

1. F1а – F2а – F3а
2. F1а – F2а – F3б

Для оцінювання якості розглянутих функцій обрана система параметрів, описана нижче.

5.3 Обґрунтування системи параметрів ПП

На підставі даних про основні функції, що повинен реалізувати програмний продукт, вимог до нього, визначаються основні параметри виробу, що будуть використані для розрахунку коефіцієнта технічного рівня.

Для того, щоб охарактеризувати програмний продукт, будемо використовувати наступні параметри: X1 – швидкодія мови програмування, X2 – об’єм пам’яті для збереження даних, X3 – час навчання моделі, X4 – потенційний об’єм програмного коду.

X1 відображає швидкодію операцій залежно від обраної мови програмування. X2 відображає об’єм пам’яті в оперативній пам’яті персонального комп’ютера, необхідний для збереження та обробки даних під час виконання програми.X3 відображає час, який витрачається на навчання моделі. X4 показує розмір програмного коду, який необхідно створити безпосередньо розробнику.

X3 та X4 – параметри функції F3

Для характеристики прототипу програмного додатку використовуємо параметри X1 – X4. На основі даних, що представлені у літературі, визначаємо мінімальні, середні отримуванні та максимально допустимі значення (табл. 5.2).

Таблиця 5.2 – Система параметрів додатку

Найменування параметру	Позначення	Значення параметру		
		Гірші	Середн	Кращі

	параметру		є	
Швидкодія мови програмування, с	X1	14	6	2
Об’єм пам’яті для збереження даних , гб	X2	38	19	8
Час навчання моделі, хв	X3	400	250	80
Об’єм програмного коду, строк	X4	3500	1900	800

За даними таблиці 5.2 будуються графічні характеристики параметрів на рис. 5.2 – 5.5.

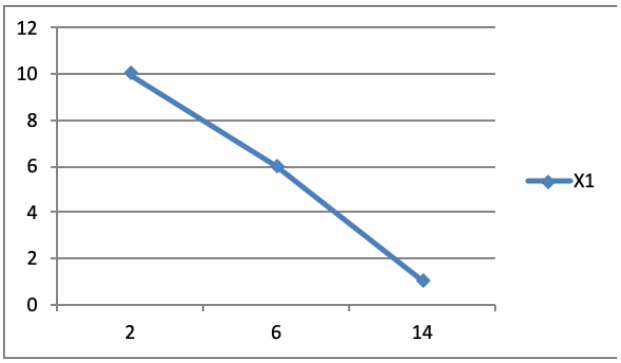


Рисунок 5.2 – Значення параметра X1 – швидкодії мови програмування

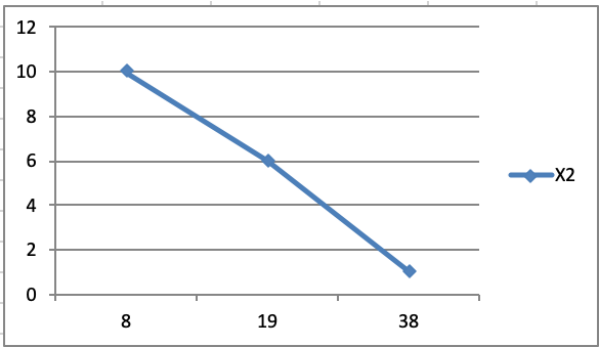


Рисунок 5.3 – Значення параметру X2 – об’єм пам’яті для збереження даних

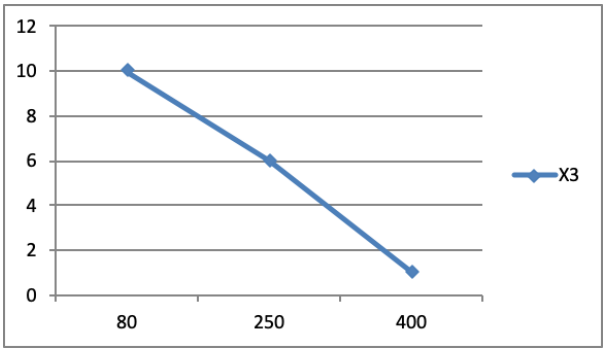


Рисунок 5.4 – Значення параметра X3 – час обробки даних алгоритмом

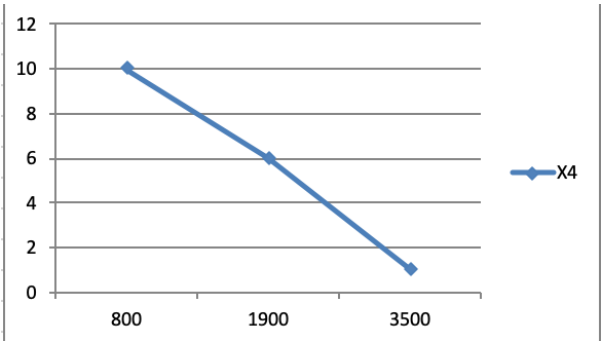


Рисунок 5.5 – Значення параметру X4 – потенційний об’єм програмного коду

Ранги варіюються від 1 до 4. Результати наведені в табл. 5.3-5.4.

Таблиця 5.3 – Результат оцінки параметрів

Параметр	Ранг параметру по оцінці експерта							Сума рангів, R_i	ВідхиленняΔ_i	Квадрат відхилення, $(\Delta_i)^2$
	1	2	3	4	5	6	7			
X1	2	2	1	2	1	2	2	12	-5,5	28,25
X2	1	1	2	1	2	1	1	9	-8,5	72,25
X3	4	3	4	4	4	3	3	25	7,5	56,25
X4	3	4	3	3	3	4	4	24	6,5	42,25
Разом	10	10	10	10	10	10	10	70	0	199

Порахуємо коефіцієнт узгодженості по формулі (5.1):

$$W = \frac{12S}{N^2(n^3 - n)} = \frac{12 * 199}{49 * (64 - 4)} = 0{,}8122 > W_k = 0{,}67 \qquad (5.1)$$

Ранжування можна вважати достовірним, тому що знайдений коефіцієнт узгодженості перевищує нормативний, котрий дорівнює 0,67.

Найвищій ранг – 4, найменший – 1.

Таблиця 5.4 – Попарне зрівняння параметрів

Параметри	Експерти							Кінцева оцінка	Числове значення
	1	2	3	4	5	6	7		
X1 та X2	>	>	<	>	<	>	>	>	1.5
X1 та X3	<	<	<	<	<	<	<	<	0.5
X1 та X4	<	<	<	<	<	<	<	<	0.5
X2 та X3	<	<	<	<	<	<	<	<	0.5
X2 та X4	<	<	<	<	<	<	<	<	0.5
X3 та X4	>	<	>	>	>	<	<	>	1.5

Розрахунок вагомості параметрів наведено в таблиці 5.5.

Як видно з таблиці 5.5, різниця значень коефіцієнтів вагомості не перевищує 2%, тому більшої кількості ітерацій не потрібно.

Таблиця 5.5 – Розрахунок вагомості параметрів

Параметри	Параметри xj				Першийкрок		Другийкрок		Третій крок	
	X1	X2	X3	X4	b_i	$К_{ві}$	b_i	$К_{ві}$	b_i	$К_{ві}$
X1	1	1,5	0,5	0,5	3,5	0,219	12,25	0,207	44,875	0,191
X2	0,5	1	0,5	0,5	2,5	0,156	9,25	0,156	34,125	0,146
X3	1,5	1,5	1	1,5	5,5	0,344	21,25	0,36	95,875	0,41
X4	1,5	1,5	0,5	1	4,5	0,28	16,25	0,275	59,125	0,253
Загалом:					16	1	59	1	234	1,00

Визначаємо рівень якості кожного варіанту виконання основних функцій окремо (табл 5.6).

Абсолютні значення параметрів X2(об’єм пам’яті для збереження даних) та X1 (швидкодія мови програмування) відповідають технічним вимогам умов функціонування даного ПП.

Абсолютне значення параметра X3 (час обробки даних) обрано не найгіршим (не максимальним), тобто це значення відповідає або варіанту а) 250 хв або варіанту б) 80хв.

Таблиця 5.6 – рівень якості кожного варіанту

Основна функція	Варіант реалізації	Абсолютне значення параметру	Бальна оцінка параметру	Коефіцієнт вагомості параметру	Коефіцієнт якості
F1	а)X1	6	6	0,191	1,15
F2	а)X2	22	5	0,146	0,73
F3	а) X3	250	6	0,41	2,46
	а) X4	1700	6,2	0,253	1,57
	б) X3	300	4,5	0,41	1,85
	б) X4	2200	5,6	0,253	1,42

Обрахуємо коефіцієнти якості кожного з варіантів розробки: -

$$К_{я1} = 1,15 + 0,73 + 2,46 + 1,57 = 5,91$$

$$К_{я2} = 1,15 + 0,73 + 1,85 + 1,42 = 5,17$$

Як видно з розрахунків, кращим є перший варіант, для якого коефіцієнт технічного рівня має найбільше значення.

5.4 Економічний аналіз варіантів розробки ПП

Для визначення вартості розробки ПП спочатку проведемо розрахунок трудомісткості.

Всі варіанти включають в себе два окремих завдання:

1. Розробка проекту програмного продукту;

2. Розробка програмної оболонки;

При цьому два варіанти мають різні додаткові завдання

Варіант один має завдання:

3.Реалізація методів аналізу.

Варіант 2 має завдання:

4.Обробка готового інтрефейсу бібліотек.

Завдання 1 за ступенем новизни відноситься до групи А, завдання 2 – до групи Б. За складністю алгоритми, які використовуються в завданні 1 належать до групи 1(алгоритми оптимізації та моделювання систем і об'єктів); а в завданні 2 – до групи 2.

Для першого завдання, виходячи із норм часу для завдань розрахункового характеру степеню новизни А та групи складності алгоритму 1, трудомісткість дорівнює: ТР =90 людино-днів. Поправочний коефіцієнт, який враховує вид нормативно-довідкової інформації для першого завдання: КП = 1.7. Поправочний коефіцієнт, який враховує складність контролю вхідної та вихідної інформації для всіх семи завдань рівний 1: КСК = 1. Оскільки при розробці першого завдання використовуються стандартні модулі, врахуємо це за допомогою коефіцієнта КСТ = 0.8. Тоді, за формулою 5.1, загальна трудомісткість програмування першого завдання дорівнює:

Т1 = 90·1.7·0.8 = 122.4 людино-днів

Проведемо аналогічні розрахунки для подальших завдань.

Для другого завдання (використовується алгоритм другої групи складності, степінь новизни Б), тобто

ТР =27 людино-днів, КП =0.9,КСК = 1,КСТ =0.8:

Т2 = 27 · 0.9 · 0.8 = 19.44 людино-днів

Для третєго завдання (використовується алгоритм другої групи складності, степінь новизни Г з використанням переміної інформації), тобто ТР =6 людино-днів, КСК = 1,КСТ =0.8:

Т2 = 6 · 0.8 = 4,8 людино-днів

Для четвертого завдання (використовується алгоритм третьої групи складності, степінь новизни Г), тобто ТР =8 людино-днів, КСК = 1,КСТ =0.85:

Т2 = 27 · 0.85 = 6,91 людино-днів

Складаємо трудомісткість відповідних завдань для кожного з обраних варіантів реалізації програми, щоб отримати їх трудомісткість:

ТІ = (122.4 + 19.44 + 4.8) · 8 = 1173,12 людино-годин

ТІІ = (122.4 + 19.44 + 6.91) · 8 = 1190 людино-годин

Найбільш високу трудомісткість має варіант ІІ.

В розробці беруть участь програміст з окладом 12000 грн. та два дата аналітика з окладом 8000грн у кожного. Визначимо зарплату за годину за формулою

Сч=(12000+8000+8000)/(3*8*21)=55,55грн

Зарплата розробників за варіантами становить:

СЗП = 55.55· 1173,12 · 1.2 = 78200.18 грн

СЗП = 55.55· 1190 · 1.2 = 79325.4 грн Відрахування на соціальний внесок становить 22%:

СВІД = СЗП · 0.22 = 78200.18 · 0.22 = 17204.04 грн

СВІД = СЗП · 0.22 = 79325.40 · 0.22 = 17451.59 грн

Так як одна ЕОМ обслуговує одного програміста з окладом 12000 грн., з коефіцієнтом зайнятості 0,2 то для однієї машини отримаємо:

СГ = 12·М·КЗ = 12 · 12000· 0,2 = 28800 грн З урахуванням додаткової заробітної плати:

СЗП =СГ· (1+ КЗ) = 28800· (1 + 0.2)=34560 грн Відрахування на соціальний внесок:

СВІД= СЗП · 0.22 = 34560· 0.22 = 7603,2 грн

Амортизаційні відрахування розраховуємо при амортизації 25% та вартості ЕОМ – 28000 грн.

СА = КТМ· КА·ЦПР = 1.15 · 0.25 · 28000 = 8050 грн,

Витрати на ремонт та профілактику розраховуємо як:

СР = КТМ·ЦПР ·КР = 1.15·28000·0.05 = 1610 грн, де КР– відсоток витрат на поточні ремонти.

Ефективний годинний фонд часу ПК за рік розраховуємо за формулою :

ТЕФ =(ДК – ДВ – ДС – ДР)·tЗ·КВ = (365 – 104 – 8 – 16)·8·0.9 = 1706.4,

Витрати на оплату електроенергії розраховуємо за формулою :

СЕЛ = ТЕФ· NC· ЦЕН =1706,4 · 0,7 · 0,2 · 1,75 = 418,07 грн,

де NC - середньо-споживча потужність приладу; КЗ - коефіцієнт зайнятості приладу;

ЦЕН - тариф за 1 КВт-годин електроенергії.

Накладні витрати розраховуємо за формулою:

СН = ЦПР·0.67 = 28000· 0,67 =18760 грн

СЕКС = 34560 + 7603.2 + 8050 + 1610 + 418,07 + 18760 = 71001.27 грн.

Собівартість однієї машино-години ЕОМ дорівнюватиме: СМ-Г = СЕКС/ ТЕФ = 71001.27 /1706.4 = 41,6 грн/час.

Оскільки в даному випадку всі роботи, які пов‘язані з розробкою програмного продукту ведуться на ЕОМ, витрати на оплату машинного часу, в залежності від обраного варіанта реалізації, складає:

СМ = СМ-Г ·Т,

СМ = 41,6 · 1328,64 = 55271.42 грн, СМ = 41,6 · 1345,52 = 55271.42 грн

Накладні витрати складають 67% від заробітної плати: СН = СЗП · 0,67,

СН = 78200.18 · 0,67 = 52394,98 грн, СН = 79325.40 · 0,67 = 53148,02 грн

Отже, вартість розробки ПП за варіантами становить: СПП = СЗП+ СВІД+ СМ +СН,

СПП = 78200.18 + 17204.04 + 55271.42 + 52394,98 = 203010.62 грн,

СПП = 79325.40 + 17451.59 + 55271.42 + 53148,02 = 205898.64 грн

5.5 Вибір кращого варіанта ПП техніко-економічного рівня

Розрахуємо коефіцієнт техніко-економічного рівня за формулою:

КТЕРj =ККj ⁄ СФj

КТЕР1 = 5,91 / 203010.62 = 0,291·10-4

КТЕР2 = 5,17 / 205898.64 = 0,251·10-4

Як бачимо, найбільш ефективним є перший варіант реалізації програми з коефіцієнтом техніко-економічного рівня КТЕР1= 0,291·10^(-4) .

5.6 Висновки

Після виконання функціонально-вартісного аналізу програмного комплексу що розроблюється, можна зробити висновок, що з альтернатив, що залишились після першого відбору двох варіантів виконання програмного комплексу оптимальним є перший варіант реалізації програмного продукту. У нього виявився найкращий показник техніко-економічного рівня якості КТЕР=$0{,}291 \cdot 10^{-4}$.

Цей варіант реалізації програмного продукту має такі параметри:

- мова програмування – Python;
- використання існуючих бібліотек;
- середовище розробки Jupyter.

Даний варіант виконання програмного комплексу дає користувачу зручний інтерфейс, непоганий функціонал і швидкодію.

ВИСНОВКИ

В даній частині було оглянуто можливість реалізації моделі згорткових нейронних мереж, в якості класифікатора рентген знімків здорової та хворої на коронавірус людини. Було досліджено принцип побудови моделі згорткової нейронної мережи, теоретичні відомості методів активації, регулярізації та ініціалізації ваг. В першому розділі було проаналізовано існуючі методи класифікації зображень, та піднята питання можливості класифікації знімків такого типу. На основі обраних для дослідження моделей, було розроблено програму (програмний продукт) який дає можливість безпосередньо передати знімок на перевірку і з високою ймовірністю отримати точний результат діагнозу.

Найкраща розроблена модель, отримана в результаті порівняння якості класифікації усіх реалізованих моделей, в перспективі робить помилку лише у

3-4 випадках із 100.

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Інститут прикладного системного аналізу
Кафедра математичних методів системного аналізу

До захисту допущено:

В.о. завідувача кафедри

_____ Оксана ТИМОЩУК

«___»_____20__ р.

Дипломна робота

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Системи та методи штучного інтелекту»
спеціальності 122 «Комп'ютерні науки та інформаційні технології»
на тему: «Моделі та методи прогнозування поширення COVID-19. Прогнозування поширення коронавірусу та аналіз впливу карантинних мір в різних країнах світу»

Виконала:
студентка IV курсу, групи КА-66
Лупаненко Софія Олександрівна _____

Консультант з економічного розділу:
доцент Шевчук Олена Анатоліївна _____

Рецензент:
доцент, к.т.н. кафедри СП ІПСА,
Безносик Олександр Юрійович _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студентка _____

Київ – 2020 року

ВСТУП

На сьогодні, проблема поширення COVID-19 є глобальною. Кожен день людство відстежує кількість нових випадків захворювань та смертей. Дані, які аналізуються в цій роботі, отримані з сайту «kaggle». Цей сайт підтримує різні набори даних. Як і відкриті так і доступні формати даних, які гарно підтримуються незалежно від інструментів. Вони зображені в часовому ряді як: дата, кількість випадків, кількість смертей та країна.

Задача цієї роботи проаналізувати часовий ряд кількості захворювань країн, знайти найкращу модель та за допомогою неї спрогнозувати кількість нових випадків.

РОЗДІЛ 6 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ ДИНАМІКИ ЗАХВОРЮВАНЬ НА COVID19 У РІЗНИХ КРАЇНАХ СВІТУ

6.1 Особливості предметної області захворювань на COVID19

В наш час відкритої інформації існують різні джерела для знаходження даних. Наприклад, ВООЗ випустила істотне оновлення інформаційної панелі COVID-19. Це дозволяє отримати доступ до поточних та достовірних даних про нові підтверджені випадки та випадки смерті в усьому світі з щоденною статистикою. Критично важливо, щоб усі країни повідомляли про свою ситуацію, адже саме це дасть змогу зробити більш точне прогнозування та аналіз ситуації сьогодні.

При моделюванні та прогнозуванні еволюційних процесів, статистичні дані представлені як часові ряди. В момент моделювання актуальною проблемою стає прогнозування подальшої поведінки часових рядів.

6.2 Постановка задачі дослідження

З практичної точки зору основні етапи побудови прогнозу часових рядів наступні:

1. Постановка задачі.

2. Попередня обробка даних, що включає видалення пропусків та видалення нетипових значень.

3. Побудова моделі: визначити фактори, що впливають; оцінка параметрів; визначення виду моделі, оцінка якості моделі.

4. Випробування (тестування прогнозів) з використанням різних моделей та ансамблів.

5. Вибір кращої моделі для прогнозування поширення.

Метою даної роботи є:

1. Порівняльний аналіз методів аналізу часових рядів:

- моделей поліноміальної регресії;

- методу опорних векторів;

- багатошарових нейронних мереж прямого розповсюдження;

- ансамблей моделей.

2. Побудова і дослідження моделей нелінійних часових рядів для прогнозування поширення коронавірусу в різних країнах світу.

3. Побудова прогнозу поширення коронавірусу на основі вибраної найкращої моделі.

4. Написання програмного продукту для побудови, дослідження моделей і знаходження прогнозу.

6.3 Висновки

В цьому розділі було розглянуто, що при моделюванні еволюційних процесів, статистичні дані представляються як часові ряди. Також було розглянуто основні етапи побудови прогнозу на основі часових рядів.

Тож, в наступному розділі основна ціль розглянути методи та моделі, які застосовують для їх прогнозування.

РОЗДІЛ 7 МОДЕЛІ ТА МЕТОДИ ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ ДАНИХ ДЛЯ ПРОГНОЗУВАННЯ ЧАСОВИХ РЯДІВ

7.1 Моделі регресії

Регресійний аналіз вивчає зв'язок між залежними та незалежними змінними, тобто прогнозними змінними через незалежні. Регресійні моделі дуже популярні в Machine leaning та використовуються для прогнозування постійного значення. Одним із поширених прикладів регресії є прогнозування ціни будинку з урахуванням його особливостей.

Лінійна регресія – це техніка, яка використовується для моделювання взаємозв'язку між однією вхідною незалежною змінною та змінною, що залежить від виходу, використовуючи лінійну модель, тобто лінію. Більш загальним випадком є мультилінійна регресія, де створюється модель для взаємозв'язку між декількома незалежними вхідними змінними та змінною залежною від виходу. Модель залишається лінійною, оскільки вихід є лінійною комбінацією вхідних змінних [1].

Найбільш популярні методи регресії використовуються для прогнозування, моделювання аналізу часових рядів та виявлення причинно-наслідкових зв’язків.

Для прогнозування доступні різні види регресійних методів. Ці методи в основному керуються трьома показниками:

- кількість незалежних змінних;
- тип залежних змінних;
- форма лінії регресії.

Формула для прогнозу лінійної регресійної моделі [2]:

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n \quad (7.1)$$

$\hat{y}$ – прогнозоване значення;

$n -$ кількість ознак ;

x_i – значення і-ої ознаки;

θ_j – j-тий параметр моделі.

Для навчання моделі лінійної регресії необхідно знайти значення θ, яке повинно зводити до мінімуму квадратний корінь середньоквадратичної помилки (рис. 7.1).

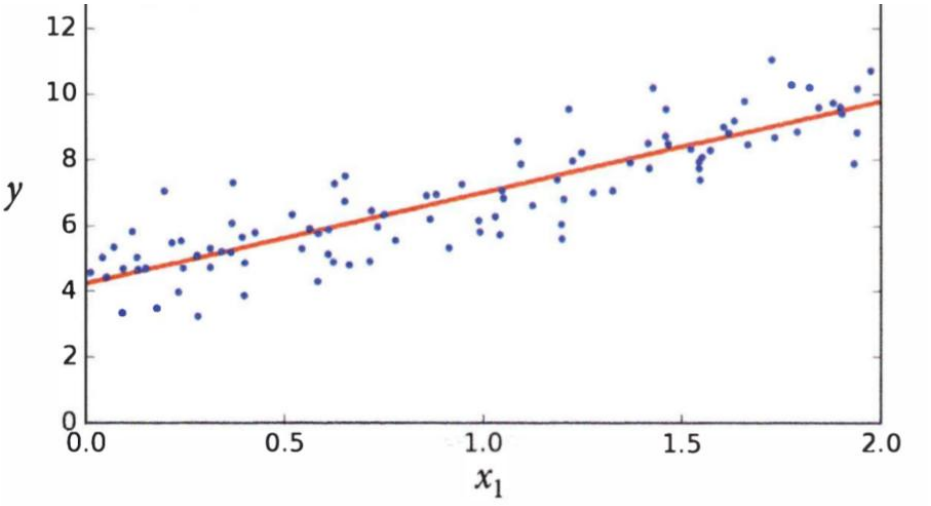


Рисунок 7.1 – Лінійна регресія

7.1.1 Регуляризовані моделі лінійної регресії

Для лінійної моделі регуляризація зазвичай застосовується шляхом обмеження ваг моделі.

Гребенева регресія (регуляторізація Тихонова) є регуляризованою версією лінійної регресії. Тільки для функцій витрат додається член регуляризації формула (7.2):

$$\alpha \sum_{i=1}^{n} \theta_i^2 \tag{7.2}$$

Гіперпараметр α керує, наскільки необхідно регуляризовати модель. Коли α = 0, гребенева регресія буде просто лінійною., при дуже великому

значенні α ваги стають близькими до нуля [2]. Тоді результатом буде рівна лінія, що проходить через середину даних (рис. 7.2).

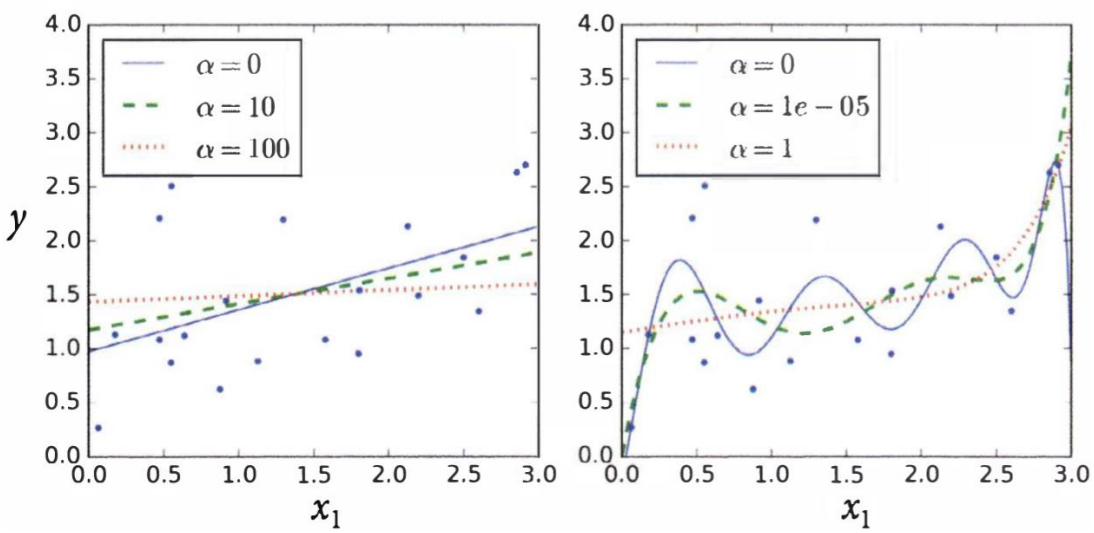


Рисунок 7.2 – Гребенева регресія

Ще однією регуляризованою версією лінійної регресії є Лассо-регресія [2]. Регресія методом найменшого абсолютного скорочення і вибору (англ. least absolute shrinkage and selectioti operator (lasso) regression). Як гребенева регресія вона додає до функції витрат член регуляризації, але замість однієї другої квадрата норми l_2 вагового вектора використовує норму l_1.

Функція втрат за формулою (7.3):

$$J(\theta) = MSE(\theta) + \alpha \sum_{i=1}^{n} |\theta_i| \qquad (7.3)$$

Важливою характеристикою лассо-регресії є те, що вона прагне повністю виключити ваги найменш важливих ознак (тобто «занульовує» їх). Наприклад, пунктирна лінія на графіку праворуч ($\alpha = 10^{-7}$) виглядає квадратичною, майже лінійною, коли всі ваги для поліноміальних ознак

високого ступеня дорівнюють нулю. Іншими словами, лассо-регресія автоматично виконує вибір ознак і випускає розріджену модель (рис. 7.3).

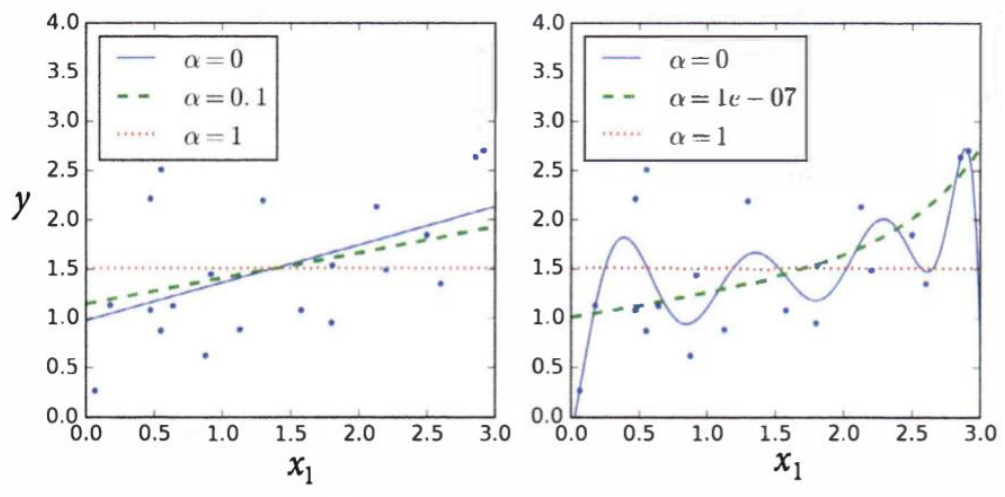


Рисунок 7.3 – Лассо-регресія

Також часто застосовується поліноміальна регресія. Цю регресію використовують як прийом для «підгонки» [1] нелінійних даних до лінійної моделі. Цей спосіб представляє додавання степенів кожної ознаки у вигляді нових ознак і подальше навчання лінійної моделі на такому розширеному їх наборі (рис. 7.4).

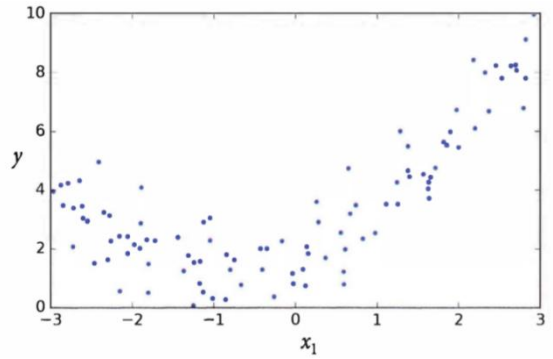


Рисунок 7.4 – Згенерований нелінійний і зашумлений набір даних.

Якщо ознак багато, то поліноміальна регресія здатна відшукати зв'язок між ними, на відміну від простої лінійної регресійної моделі (рис. 7.5).

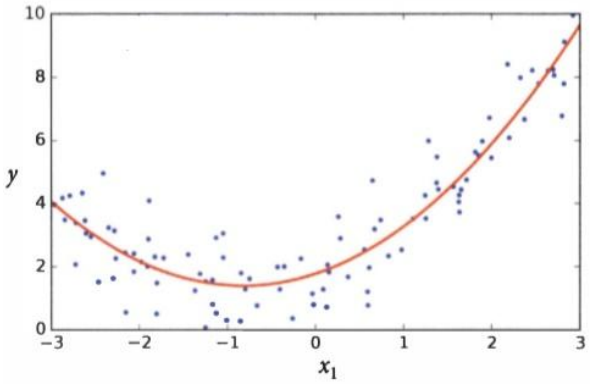


Рисунок 7.5 – модель поліноміальної регресійної моделі.

7.1.2 Узагальнена регресія

Узагальнені лінійні моделі (GLM) розширюють лінійні моделі двома способами.

По-перше, передбачувані значення $\hat{y}$ пов'язані з лінійною комбінацією вхідних змінних X через функцію зворотного зв'язку h за фомулою:

$$\hat{y}(w, X) = h(Xw) \tag{7.4}$$

По-друге, функція втрат у квадраті замінюється на відхилення одиниці d розподілу в родині експонентів, тобто модель репродуктивної експоненціальної дисперсії (EDM) [3].

Проблема мінімізації за формулою:

$$\min_w \frac{1}{2n_{samples}} \sum_i d(y_i, \hat{y}_i) + \frac{\alpha}{2} \|w\|_2, \tag{7.5}$$

де α – штраф за регуляризацію l_2. Коли надаються вагові зразки, середнє значення стає середньозваженим.

Вибір розподілу (рис. 7.6) залежить від проблеми:

- Якщо цільові значення **y** є підрахунком (невід'ємне ціле число) або відносними частотами (негативними), то можна використовувати відхилення Пуассона з логічним посиланням.
- Якщо цільові значення позитивно оцінені та перекошені, то можна спробувати відхилення Гамми.
- Якщо цільові значення здаються більш важкими, ніж розподіл гамма, ви можете спробувати відмінні гауссові відхилення.

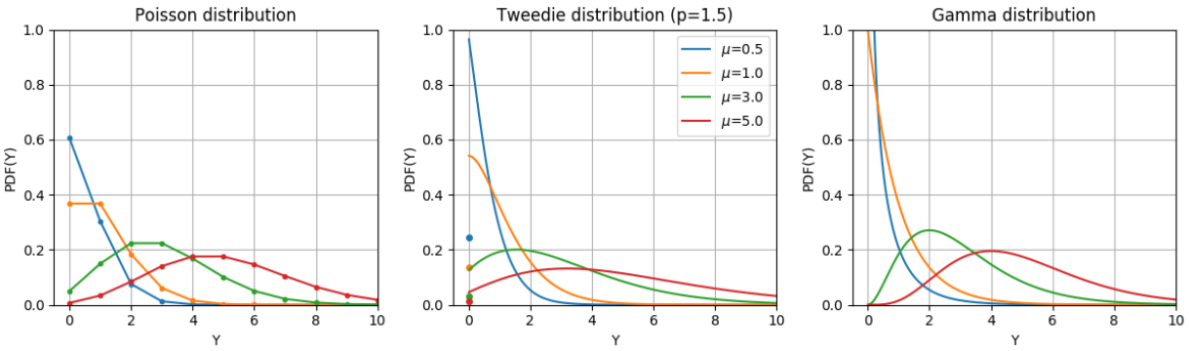


Рисунок 7.6 – Функція щільності ймовірності випадкової величини Y після розподілу Пуассона, Твіді (потужність = 1,5) та гамма з різними середніми значеннями (μ).

7.2 Метод опорних векторів

Метод опорних векторів (Sipport Vector Machine – SVM) – це дуже потужна універсальна модель машинного навчання, здатна виконувати лінійну або нелінійну класифікацію, регресію і також виявлення викидів. Вона є однією з найпопулярніших моделей в машинному навчанні. Методи SVM особливо добре підходять для класифікації складних, але невеликих або середніх наборів даних.

Також вони чутливі до масштабів ознак, як можна бачити на рисунку 7.7. Графік зліва має масштаб по вертикалі, набагато перевищує масштаб по горизонталі, тому найширша смуга близька до горизонталі. Після масштабування ознак межа рішень виглядає набагато краще (на графіку праворуч).

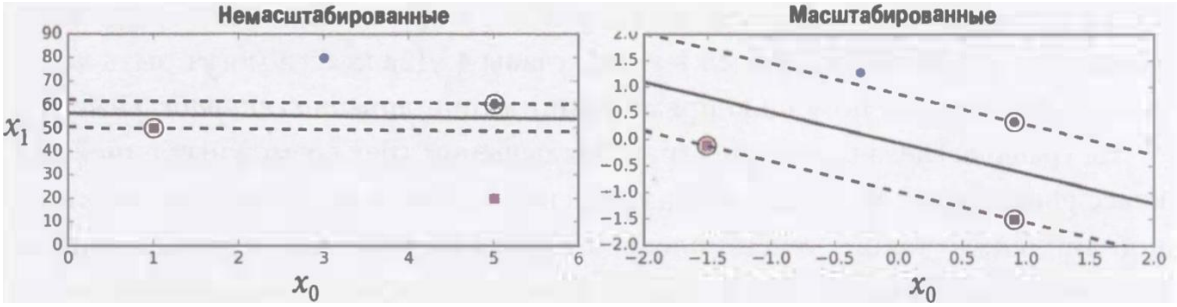


Рисунок 7.7 – Чутливість до масштабування

Як згадувалося раніше, алгоритм SVM досить універсальний, адже він підтримує не тільки лінійну і нелінійну класифікацію, а також лінійну і нелінійну регресію.

Прийом полягає в інвертуванні мети. Тобто замість спроби пристосуватися до найширшої з можливих смуг між двома класами, одночасно обмежуючи порушення зазору, регресія SVM пробує вмістити якомога більше зразків на смузі поряд з обмеженням порушень зазору, тобто зразків поза смугою.

Ширина смуги залежить від гіперпараметром – ε. На рисунку 2.8 показані дві моделі лінійної регресії SVM, котрі навчені на випадкових лінійних даних: з вузьким зазором (ε = 0.5) і з широким зазором (ε = 1.5) [2].

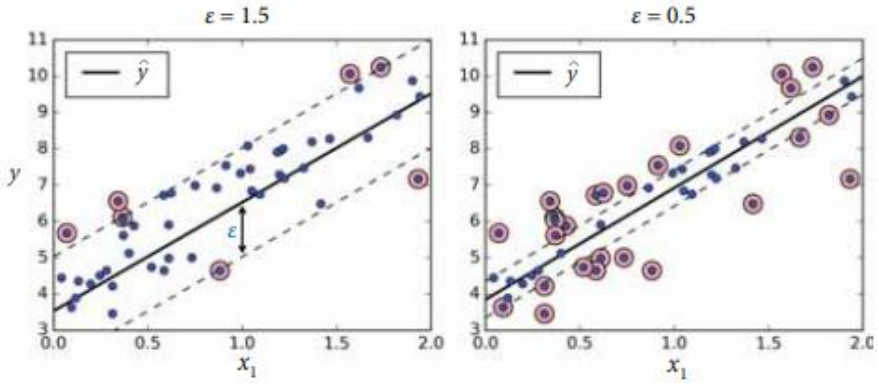


Рисунок 7.8 – Регресія SVM

Для вирішення задач нелінійної регресії можна застосовувати параметричну редуційовану (kernelized) модель SVM. На рисунку 7.9 демонструється регресія SVM на випадковому квадратичному навчальному наборі, котра використовує поліноміальне ядро 2-го порядку. На графіку

зліва вироблялося меньше регуляризації, тобто велике значення C, а на графіку праворуч, навпаки, більше регуляризації, тобто невелике значення C.

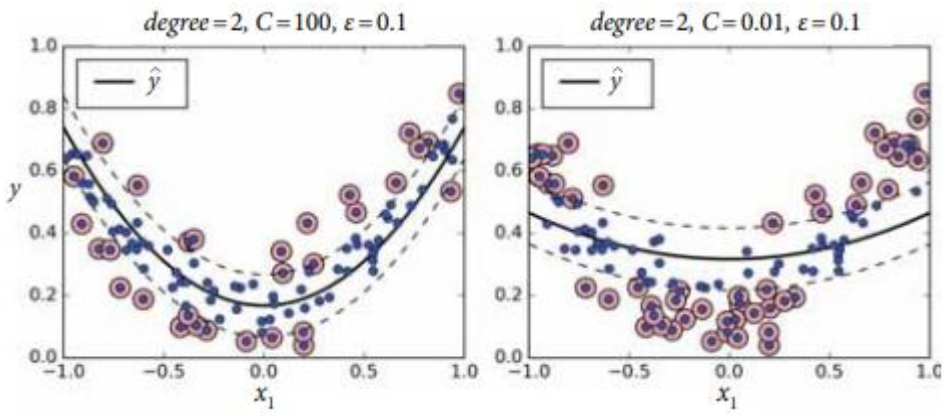


Рисунок 7.9 – Регресія SVM, яка застосовує поліноміальне ядро 2-го порядку

7.3 Нейронні мережі прямого розповсюдження

7.3.1 Побудова архітектури мережі

Розглянемо персептрон Розенблатта – одношарову нейронну мережу прямого поширення, яку показано на рисунку 2.10 [4]. Ця мережа лежить в основі сучасних нейронних мереж.

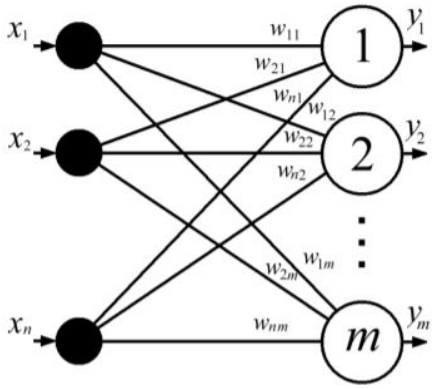


Рисунок 7.10 – Одношарова нейрона мережа

Спроби застосувати одношарові нейронні мережі для розв’язування широкого кола задач завдали труднощів, це пов’язано з проблемою лінійної роздільності. Вирішенням цієї проблеми стало застосування багатошарових ШНМ, що нагадують багатошарові структури.

Розглянемо ієрархічну структуру, зображену на рисунку 2.11. Вона складається з прихованого шару, що містить m нейронів, та вихідного шару, що містить k нейронів. На вхід прихованого ширу подається вхідний вектор сигналів. Ця модель вважається узагальненою структурою багатошарової нейронної мережі прямого поширення [4].

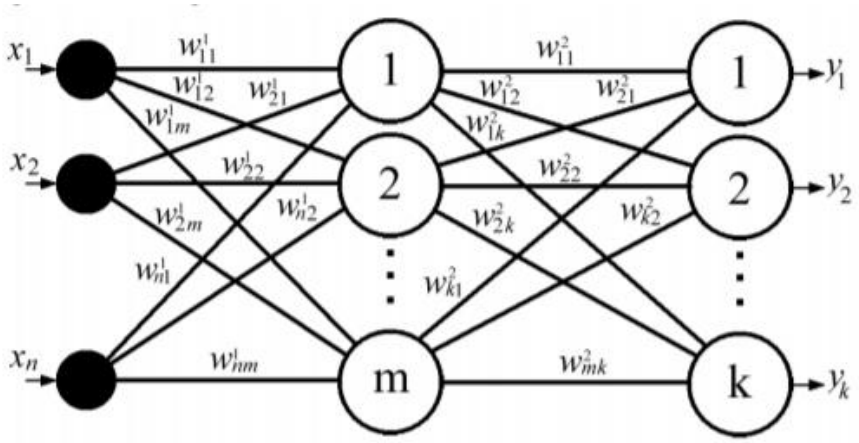


Рисунок 7.11 – багатошарова нейрона мережа

Побудова моделі мережі складається з наступних кроків [Недашківська, конспект]:

1. Вибір початкової архітектури мережі. Наприклад, будують один проміжний шар з кількістю елементів в ньому, що дорівнює половині сумі числа входів і числа виходів.
2. Проведення експериментів з різними архітектурами. При цьому краща мережа за вибраною метрикою запам'ятовується. Для кожної архітектури проводиться кілька експериментів, з метою уникнення помилкового результату внаслідок потрапляння процесу навчання у локальний мінімум.
3. Якщо в черговому експерименті спостерігається неповне навчання, то додати один або декілька нейронів в проміжні шари. Також можна додати ще один проміжний шар.
4. Якщо маємо в результаті перенавчання (контрольна помилка стала рости), то треба спробувати видалити кілька прихованих елементів (а можливо і шарів).

7.3.2 Вибір методу оптимізації

Один з підходів до оптимізації роботи нейронної мережі – це виконати регуляризацію синаптичних коефіцієнтів мережі. Інший підхід включає оптимізацію архітектури нейронної мережі (НМ) з метою забезпечення більш ефективного функціонування в більшому просторі порівняно з тим, що використовувався в процесі навчання цієї мережі. Найбільш простий підхід до підвищення ефективності функціонування НМ полягає у безпосередньому застосуванні знань про вхідні дані на етапах побудови архітектури та навчання [5]. Такий підхід, як правило, є недосяжним в основному з двох причин [5]:

- недостатність знань про всю можливу множину вхідних даних;

- великий об’єм роботи, яку необхідно виконати у випадку великих масивів вхідних даних.

На сьогоднішній день вважається, що жоден із сучасних методів оптимізації архітектури не може гарантувати досягнення найкращого результату для кожного конкретного випадку. Сучасні методи базуються на досягненні компромісу між часом навчання мережі, кількістю елементів мережі та ефективністю узагальнення. Критерієм оптимізації слугує мінімізація цільової функції, в ролі якої найчастіше використовується середньоквадратична похибка, що поділяється на навчальну та реальну [6].

Реальною похибкою називають похибку, з якою функціонує мережа в умовах реальних даних. Відомо, що реальна похибка зазвичай перевищує навчальну. Тому для визначення ступеня коректності роботи мережі важливо оцінити реальну похибку. Для знаходження оцінки реальної похибки використовується підхід, який полягає у використанні оцінки на незалежній підмножині. Незалежна підмножина має включати діапазон зміни даних, який планується використовувати в умовах реального функціонування, бути сформованою до проведення навчання мереж і не повинна використовуватися в майбутньому [Недашківська, конспект]. Такий підхід

характерний для НМ, які працюють з неперервними потоками звукової, відео або телекомунікаційної інформації. Коли вхідний масив даних обмежений, використовують метод багаторазової оцінки [6]. Цей метод полягає в реалізації об’єднаної процедури навчання та визначення її реальної похибки.

Розглянемо метод попередньої зупинки. Ретельна підгонка параметрів ШНМ на фіксованій навчальній вибірці може призвести до надто точної настройки на особливості конкретних даних, що часто викликає збільшення значення реальної похибки. Для виходу з цієї ситуації використовують зупинку процесу навчання до того моменту, поки значення реальної похибки не почне зростати.

Для оптимізації архітектури нейронних мереж також використовують різні модифікації наведених вище методів. Наприклад, через певні визначені інтервали обчислюють навчальну та реальну похибки. Зупинку виконують за наявності тенденції до зростання реальної похибки. Але вважається, що таке визначення моменту зупинки може призвести до попадання у локальний мінімум, якщо у вхідному сигналі присутній шум або значна нелінійність [6].

Для того, щоб уникнути попадання в локальний мінімум, оцінку реальної похибки виконують на підмножині параметрів [Недашківська, конспект]. Існують також інші методики, що базуються на встановленні порогу чутливості, коли умова попередньої зупинки – це перевищення реальною похибкою значення цього порогу. Конкретна величина такого порога залежить від вхідних даних, а також від архітектури нейронної мережі, тому її визначають у результаті практичної роботи з НМ.

7.3.3 Методи регуляризації параметрів нейронної мережі

Регуляризація за нормами L1 і L2.

L1-регуляризація параметрів моделі w визначається за формулою:

$$\Omega(\theta) = \|w\|_1 = \sum_i |w_i| \tag{7.6}$$

Тобто, як сума абсолютних величин окремих параметрів.

Розглянемо вплив L1-регуляризації на просту модель лінійної регресії без параметра зсуву та які відмінності між двома видами регуляризації. Сила регуляризації контролюється шляхом множення штрафу на позитивний гіперпараметр α. Таким чином, регулярізірованная цільова функція J (w; X, y) описується формулою:

$$\tilde{J}(w; X, y) = \alpha\|w\|_1 + J(w; X, y) \tag{7.7}$$

Її градієнт дорівнює:

$$\nabla_w \tilde{J}(w; X, y) = \alpha\, \text{sign}(w) + \nabla_w J(w; X, y), \tag{7.8}$$

де sign (w) означає, що функція sign застосовується до кожного елементу w.

З формули (7.8) відразу видно, що ефект L1-регуляризації зовсім не такий, як L2-регуляризації. Тепер внесок регуляризації в градієнт вже не масштабується лінійно з ростом кожного w_i, а описується постійним доданком, знак якого збігається з sign (w_i). Одним із наслідків є той факт, що результатом вже не буде витончених алгебраїчних виразів кадратичної апроксимації J (X; y, w), як у випадку L2-регуляризації [3].

У простій лінійній моделі є квадратична функція вартості, котру можна представити рядом Тейлора. Можна замість цього вважати, що це перші члени ряду Тейлора, аппроксимуючи функцію вартості більш складної моделі. Градієнт в цій конфігурації дорівнює:

$$\nabla_w \hat{J}(w) = H(w - w^*), \tag{7.9}$$

де H – знову матриця Гессе J щодо w, обчислена в точці w *.

Оскільки штраф за нормою L1 не допускає простого алгебраїчного виразу в разі загального вигляду, то будемо вважати матрицю Гессе діагональною.

$H = \text{diag}([H_1, 1, \ldots, H_n, n])$, де всі $H_i, i > 0$ – це припущення справедливо, якщо дані для задачі лінійної регресії були піддані попередній обробці для усунення кореляції між вхідними ознаками, наприклад методом головних компонент.

Квадратичную апроксимацію L1-регуляризованої цільової функції можна представити у вигляді суми за параметрами:

$$\hat{J}(w; X, y) = j(w^*; X, y) + \sum_i \left[\frac{1}{2} H_{ij}\left(w_i - w_i^*\right)^2 + \alpha|w_i|\right] \qquad (7.10)$$

У завданні мінімізації наближеної функції вартості є аналітичне рішення (для кожного вимірювання i) виду:

$$w_i = \text{sign}(w_i^*)\max\left\{|w_i^*| - \frac{\alpha}{H_{ij}}, 0\right\} \qquad (7.11)$$

1) $w_i^* \leq \alpha / H_i$, i. Тоді оптимальне значення w_i для регуляризованої цільової

функції буде просто $w_i = 0$. Причина в тому, що внесок J (w; X, y) в регуляторну цільову функцію переважується в напрямку i. L1 – регуляризацією, яка зрушує значення w_i в нуль;

2) $w_i^* > \alpha / H_i$, i.

Тоді регуляризація не зрушує оптимальне значення w_i в нуль, а просто зміщує його в цьому напрямку на відстань α / H_i, i.

Аналогічне міркування проходить, коли $w_i^*<0$, тільки L1-штраф збільшує w_i на α / H_i, і чи обертає в 0.

У порівнянні з L2-регуляризації, L1-регуляризація дає більш розріджене рішення. Під розрідженістю розуміється той факт, що у деяких параметрів оптимальне значення дорівнює 0. Розрідженість L1-регуляризації являється якісною відмінністю від поведінки L2-регуляризації. Рівняння (7.12) дає рішення $\widetilde{w}$ для L2-регуляризації. Застосувавши до нього припущення про діагональний і позитивної визначеності гессіан H, яке ми прийняли для аналізу L1 регулярізації, знайдемо, що:

$$\widetilde{w_i} = \frac{H_{ij}}{H_{ij} + \alpha} w_i^* \tag{7.12}$$

Якщо w* не дорівнює 0, то і $\widetilde{w}$ не буде рівним нулю. Це означає, що L2-регуляризація не призводить до розрідженості параметрів, тоді як у разі L1-регуляризації це можливо, якщо α досить має досить велике значення.

А L2-регулярізация як і L1 регулярізація [2], є одною з найпростіших та використовуємим методом регулярізації, який часто називають методом зменшення ваг.

Мета такої стратегії регуляризації – вибирати ваги, близькі до початку координат, за рахунок додавання до цільової функції члена регуляризації:

$$\Omega(\theta) = \frac{1}{2}\|w\| \tag{7.13}$$

В наукових спільнотах регуляризацію по нормі L2 називають також гребеневою регресією, або регуляризацією Тихонова.

Отримати якісне уявлення про поведінку регуляризації методом зменшення ваг можна, визначивши градієнт регуляризованої цільової функції. Для спрощення опустимо параметр зсуву. Будемо вважати, що θ збігається з w. Повна цільова функція в такій моделі має вигляд:

$$\tilde{J}(w; X, y) = (\alpha/2)w^{\top}w + J(w; X, y), \tag{7.14}$$

а градієнт за параметрами:

$$\nabla_w \tilde{J}(w; X, y) = \alpha w + \nabla_w J(w; X, y) \tag{7.15}$$

Один крок оновлення ваг з метою зменшення градієнта має вигляд:

$$w \leftarrow w - \varepsilon(\alpha w + \nabla_w J(w; X, y)) \tag{7.16}$$

Те ж саме можна переписати у вигляді:

$$w \leftarrow (1 - \varepsilon\alpha)w - \varepsilon\nabla_w J(w; X, y) \tag{7.17}$$

7.4 Ансамблі моделей

Методи ансамблевого навчання ґрунтуються на гіпотезі, що комбінація декількох моделей разом може створити більш потужну модель. Тож приказка «сила – це єдність» саме про цей випадок.

Ансамблеве навчання – це парадигма машинного навчання, де декілька моделей поєднуються для отримання кращих результатів та вирішують спільну проблему [4]. Основна ідея полягає в тому, щоб спробувати зменшити дисперсію слабких моделей, комбінуючи декілька з них разом, щоб створити ансамблеву модель, котра досягає кращих результатів.

Для того, щоб створити ансамблевий метод навчання, спочатку потрібно вибрати базові моделі для агрегації. Більшу частину часу використовується єдиний алгоритм базового навчання. Отримана модель ансамблю називається «однорідною» (рис. 7.13).

Рисунок 7.13 – Однорідний ансамбль

Однак існують деякі методи, які використовують різні типи алгоритмів базового навчання: деякі неоднорідні слабкі моделі потім об'єднуються у модель різнорідних ансамблів, наприклад нейронних мереж, дерев рішень, регресійних моделей тощо (рис. 7.14).

Рисунок 7.14 – Ансамбль з моделей різного типу

Важливим моментом є те, що вибір слабких моделей повинен бути узгодженим із способом узагальнення цих моделей. Якщо вибирати базові моделі з низьким ухилом, але з великою дисперсією, це має бути з агрегуючим методом, який має тенденцію до зменшення дисперсії, а якщо вибирати базові моделі з низькою дисперсією, але з великим зміщенням, то це має бути з агрегуючим методом, який прагне зменшити зміщення.

7.4.1 Стекінг

Його ідея полягає в тому, щоб навчити кілька різних слабких моделей [5] та поєднати їх, навчаючи мета-модель для виведення прогнозів на основі множинних прогнозів цих моделей. Отже, нам потрібно визначити дві речі для побудови моделі: моделі, які ми хочемо застосовувати, та мета-моделі, які їх поєднує.

Наприклад, в задачах класифікації в якості слабкої моделі вибирають класифікатор KNN, логістичну регресію або SVM. Тоді нейронна мережа сприймає результати і навчиться повертати на її основі остаточні прогнози.

Отже, припустимо, що ми хочемо вмістити ансамбль стекінгу, який складається з L моделей. Потім треба виконати наступні кроки:

- розділити дані тренувань у два рази;
- вибирайте L слабких учнів і пристосовуйте їх до даних першого згину;
- для кожного з моделей, що склалися, зробіть прогнози для спостережень у другій частині;
- розмістити мета-модель на другій складці, використовуючи прогнози, зроблені слабкими моделями як вхідні дані.
-

7.4.2 Бустінг

Бустінг – це техніка [5], яка полягає в підгонці послідовно декількох слабких моделей адаптивним способом. Тобто кожна модель послідовно надає більше значення спостереженням у наборі даних. Інтуїтивно, кожна нова модель навчається більш складніше, щоб отримати в кінці процесу сильну модель з меншими похибками. Бустінг також може використовуватися як для регресії, так і для проблем класифікації.

Наприклад, якщо ми хочемо використовувати дерева в якості базових моделей, ми будемо вибирати більшу частину дерев з дрібними рішеннями лише з декількома глибинами. Ще одна важлива причина, яка мотивує використання моделей з низькою дисперсією, але з високим ухилом це те, що моделі, як правило, є менш обчислювально дорогими для встановлення (кілька ступенів свободи при параметризації). Оскільки обчислення для відповідності різних моделей не можна робити паралельно (на відміну від бегінгу), це може стати занадто дорогим, щоб послідовно підходити до декількох складних моделей.

7.4.3 Беггінг

Беггінг є технологією “збурення та комбінування” [6]. Під час навчання моделі, незалежно від того, чи маємо ми справу з класифікацією чи регресійною проблемою, ми отримуємо функцію, яка приймає вхідні дані, повертає результат і визначається стосовно навчального набору даних.

Ідея бегінгу полягає в побудові декількох альтернативних моделей на змінених даних з наступним комбінування результатів і внесенні змін випадкового характеру в навчальні дані.

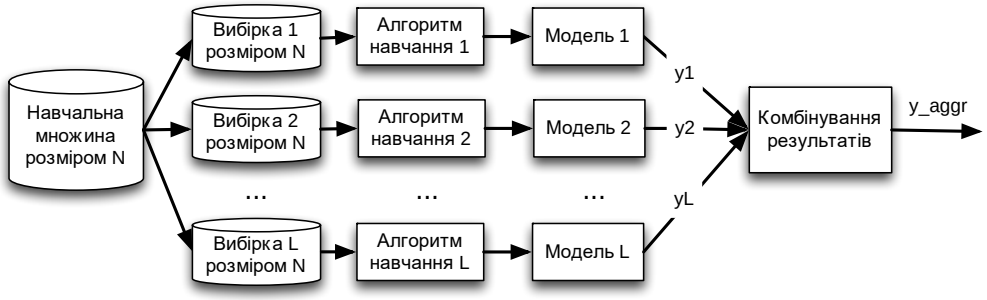


Рисунок 7.15 – Беггінг

Етапи:

- Формування вибірок однакового розміру випадковим чином на основі навчальної множини;
- Побудова моделі на основі кожної вибірки;
- Комбінування результатів.

7.5 Висновки

В цьому розділі приведено регресійні моделі, метод опорних векторів, методи регуляризації параметрів нейронної мережі та ансамблі моделей, за допомогою яких можна спрогнозувати поширення пандемії.

РОЗДІЛ 8 РЕЗУЛЬТАТИ ПРОГНОЗУВАННЯ ПОШИРЕННЯ КОРОНАВІРУСУ ТА АНАЛІЗУ ВПЛИВУ КАРАНТИННИХ МІР В РІЗНИХ КРАЇНАХ СВІТУ

8.1 Обґрунтування вибору бібліотеки Scikit-Learn Python

Вибір бібліотеки Scikit-Learn оснований на тому, що вона надає широкий вибір алгоритмів. Основною її перевагою є поєднання в собі декількох поширених математичних бібліотек. Також цю бібліотеку використовують як і новачки, бо вона має докладну документацію, так і для промислових систем, котрі застосовують алгоритми класичного машинного навчання для досліджень.

Для своєї роботи, Scikit-Learn використовує такі популярні бібліотеки як:

- «Pandas»: для обробки, маніпуляції і аналізу даних;
- «SciPy»: для науково-технічних обчислень;
- «SymPy»: для символьної математики;
- «Matplotlib»: для візуалізації даних;

Базовий список методів реалізації бібліотеки:

- Лінійні моделі;
- Метричні моделі;
- Дерева рішень;
- Ансамблеві методи;
- Нейронні мережі;
- SVM;
- Наївний;
- PCA;
- t-SNE;
- K-середніх;

- Крос-валідація;
- Grid Search.

З цих методів мене цікавлять:

1. Ансамблеві методи бустінг, беггінг, стекінг, засновані на деревах рішень, які комбінують множину дерев, і внаслідок цього підвищують їх якість роботи. Також ці методи дозволяють проводити відбір ознак.
2. Нейронні мережі – комплексний нелінійний метод для задач регресії.
3. SVM– нелінійний метод для задач регресії, який навчається визначати межі прийняття рішень.

Тож, вибір бібліотеки Scikit-Learn досить влучний для дослідження побудови найкращої моделі для прогнозування часових рядів.

8.2 Опис оцінки якості прогнозування часових рядів

Критерій оцінки якості виду $V = V(X, \widehat{X})$. Для оцінки якості використовую два критерія:

1. Визначає найкраще співпадіння значення максимуму прогнозу, формула (8.1):

$$V = V(X, \widehat{X}) = |\max\{x(t_1), \dots, x(t_k)\} - \max\{\hat{x}(t_1), \dots, \hat{x}(t_k)\}| \quad (8.1)$$

2. Визначає співпадіння значення такту $x(t_k)$ з максимальним прогнозованим значенням, формула (8.2):

$$V = V(X, \widehat{X}) = \\ = |\operatorname{argmax}\{x(t_1), \dots, x(t_k)\} \\ - \operatorname{argmax}\{\hat{x}(t_1), \dots, \hat{x}(t_k)\}| \quad (8.2)$$

Для оцінки якості прогнозу, котрий створений із застосуванням формалізованих методів, загально прийнято застосовувати подібні характеристики для достовірності прогнозу. Під достовірністю прогнозних розрахунків розуміється поведінка об'єкта прогнозування в часі. Достовірність прогнозу обумовлюється ймовірністю реалізації прогнозу для встановленого варіанту або довірчого інтервалу.

Для оцінки працездатності моделей використовується декілька критеріїв якості: R^2, стійкість та відсоток викиду.

Ефективність Неша-Сатклiффа, або R^2, є найбільш часто використовуваним критерієм.

$$R^2 = 1 - \frac{\sum_{k=1}^{n} \left(y_0^k - y^k\right)^2}{\sum_{k=1}^{n} \left(y_0^k - \overline{y_0}\right)^2} \qquad (8.3)$$

Результати R^2, котрі ближче до одиниці є кращими. Значення $R^2 > 0.75$ – відповідає нормі оцінки прогнозу.

8.3 Побудова і дослідження моделей нелінійних часових рядів для прогнозування поширення коронавірусу в різних країнах світу

8.3.1 Дослідження нелінійних часових рядів

На рисунках 8.1 – 8.5 представлений графічно часовий ряд випадків захворювань. На графіку зліва зображена кількість захворювань щоденно, а на графіку справа сумарна кількість випадків за весь час певної країни.

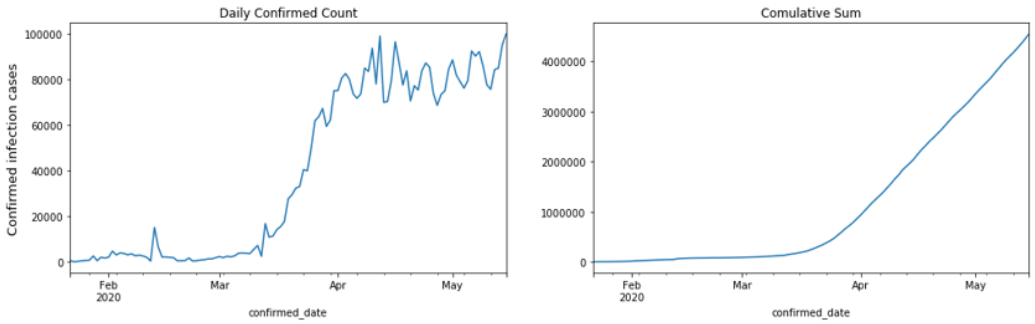


Рисунок 8.1 – Графічно представлений часовий ряд випадків захворювань у світі

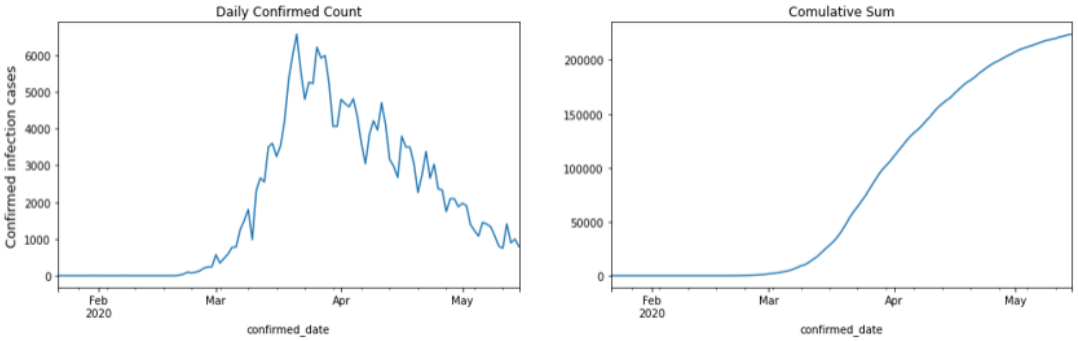


Рисунок 8.2 – Графічно представлений часовий ряд випадків захворювань в Італії

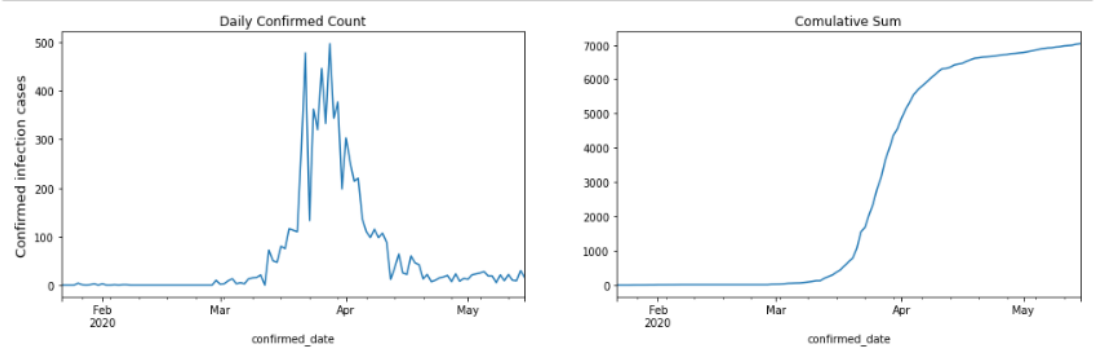


Рисунок 8.3 – Графічно представлений часовий ряд випадків захворювань в Австралії

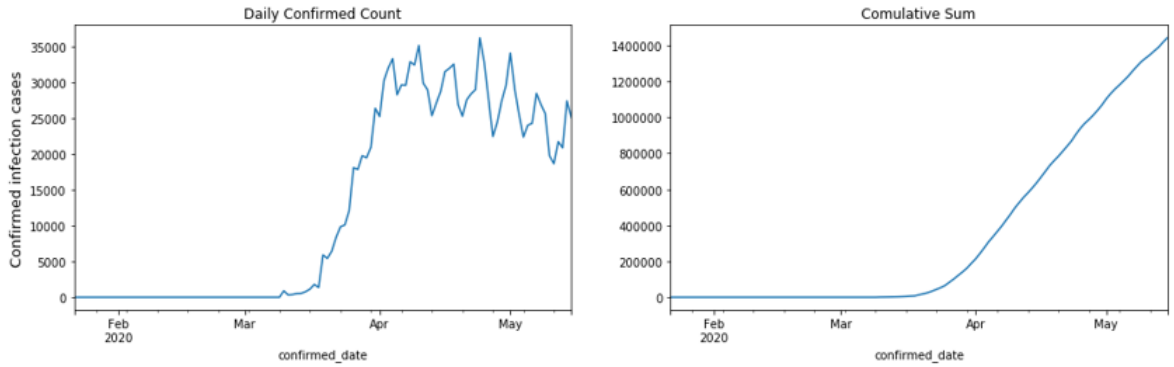


Рисунок 8.4 – Графічно представлений часовий ряд випадків захворювань в США

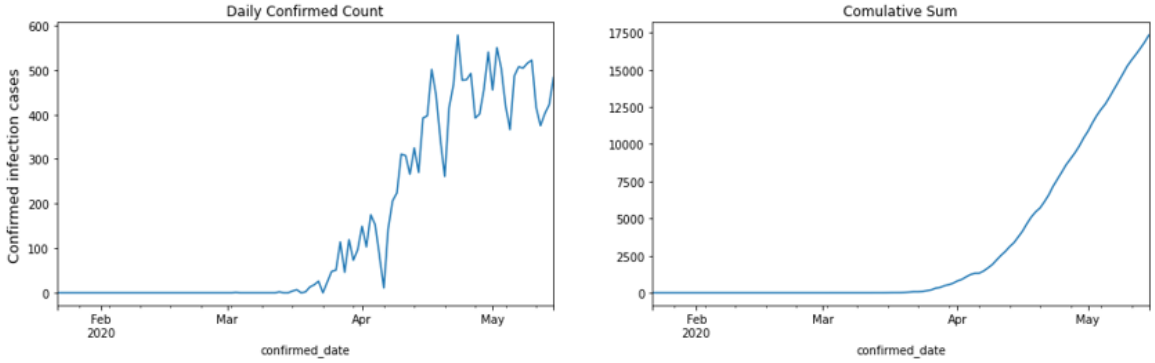


Рисунок 8.5 – Часовий ряд випадків захворювань в Україні

Проаналізувавши часовий ряд, можна зробити висновок як саме робити прогноз. Перед тим як навчати модель треба поділити вибірку на навчальну та тренувальну (рис. 8.6). В даному випадку навчальна вибірка – це 30 днів щоденної кількості випадків захворювань, за тиждень до піку випадків захворювань кожної країни. А після навчання моделі прогнозується щоденна кількість уражених на 7 днів.

```
def train_and_test(df):
    pick_date = df[(df['daily_count'] == df['daily_count'].max())].index[0]
    test_start_date = pick_date -  timedelta(days=7)
    train_start_date = test_start_date -  timedelta(days=30)
    train = [train_start_date,(test_start_date-timedelta(days=1))]
    test = [test_start_date,pick_date]
    return train, test
```

Рисунок 8.6 – Розподіл на тестову та тренувальну вибірку

8.3.2 Результати і аналіз на основі багатошарового персептрона для задачі регресії

Проаналізуємо результати прогнозу на основі багатошарового персептрона для задачі регресії за допомогою методів розрахунку ваг «lbfgs» та «adam».

Результати прогнозу на рисунках 8.7 – 8.14 на основі методу «lbfgs» розглянемо на прикладі – США, Росії, Бельгії, України.

```
Predicted data:
2020-04-17    694145
2020-04-18    722378
2020-04-19    750612
2020-04-20    778845
2020-04-21    807078
2020-04-22    835312
2020-04-23    863545
dtype: int32
Real data:
confirmed_date
2020-04-17    699541.0
2020-04-18    732031.0
2020-04-19    758920.0
2020-04-20    784160.0
2020-04-21    811699.0
2020-04-22    840054.0
```

Рисунок 8.7 – Результати прогнозу та існуючих даних для США

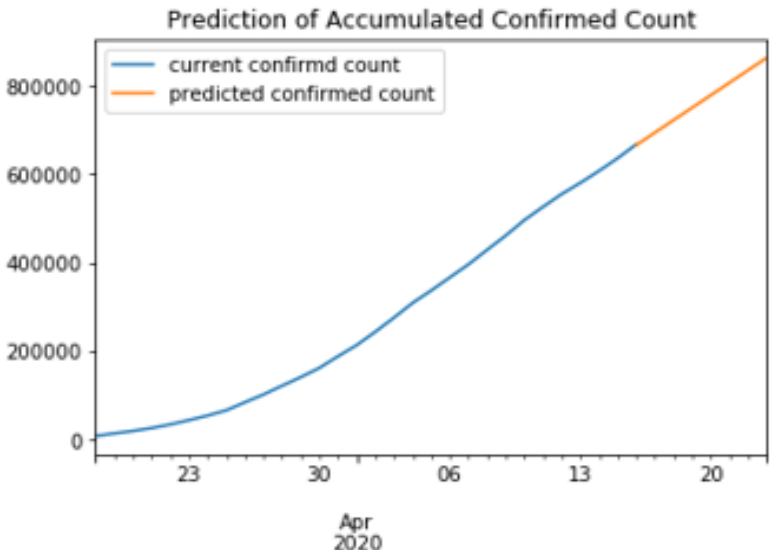


Рисунок 8.8 – Графічно представлений прогноз для США на 7 днів

```
Predicted data:
2020-05-04    145331
2020-05-05    155971
2020-05-06    166611
2020-05-07    177252
2020-05-08    187892
2020-05-09    198533
2020-05-10    209173
dtype: int32
Real data:
confirmed_date
2020-05-04    145268.0
2020-05-05    155370.0
2020-05-06    165929.0
2020-05-07    177160.0
2020-05-08    187859.0
2020-05-09    198676.0
2020-05-10    209688.0
```

Рисунок 8.9 – Результати прогнозу та існуючих даних для Росії

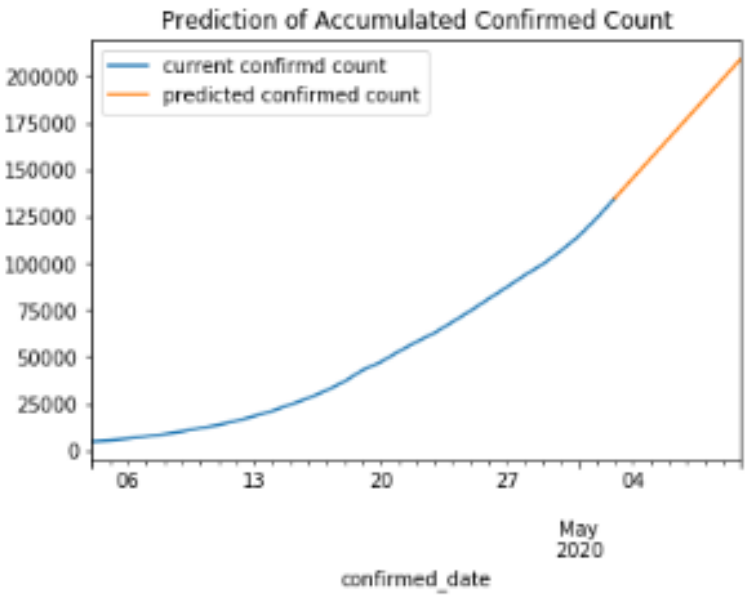


Рисунок 8.10 – Графічно представлений прогноз для Росії на 7 днів

```
Predicted data:
2020-04-08    23521
2020-04-09    24845
2020-04-10    26168
2020-04-11    27492
2020-04-12    28815
2020-04-13    30139
2020-04-14    31463
dtype: int32
Real data:
confirmed_date
2020-04-08    23403.0
2020-04-09    24983.0
2020-04-10    26667.0
2020-04-11    28018.0
2020-04-12    29647.0
2020-04-13    30589.0
2020-04-14    31119.0
```

Рисунок 8.11 – Результату прогнозу та існуючих даних для Бельгії

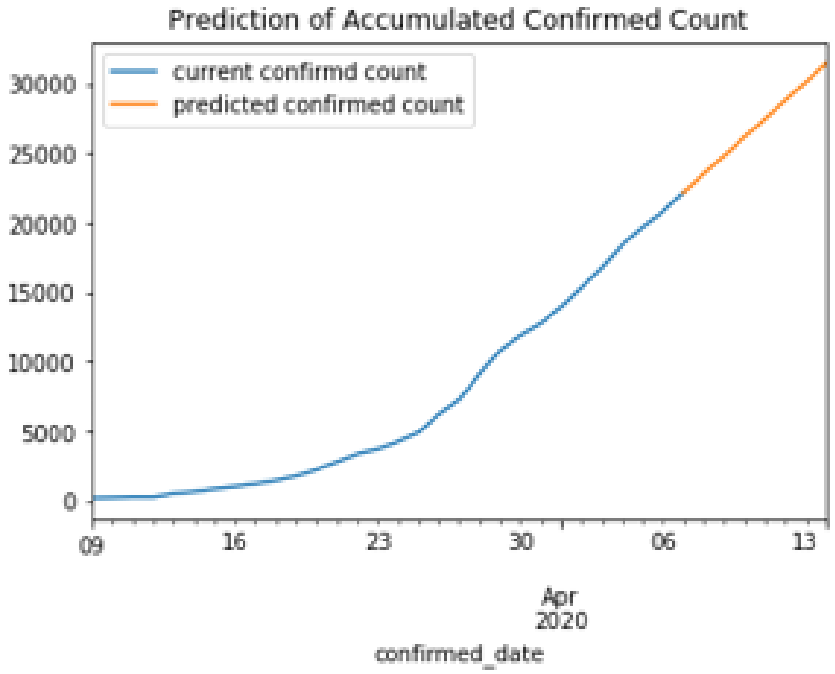


Рисунок 8.12– Графічно представлений прогноз для Бельгії на 7 днів

```
Predicted data:
2020-04-16    4065
2020-04-17    4420
2020-04-18    4785
2020-04-19    5151
2020-04-20    5518
2020-04-21    5884
2020-04-22    6250
dtype: int32
Real data:
confirmed_date
2020-04-16    4161.0
2020-04-17    4662.0
2020-04-18    5106.0
2020-04-19    5449.0
2020-04-20    5710.0
2020-04-21    6125.0
2020-04-22    6592.0
```

Рисунок 8.13 – Результату прогнозу та існуючих даних для України

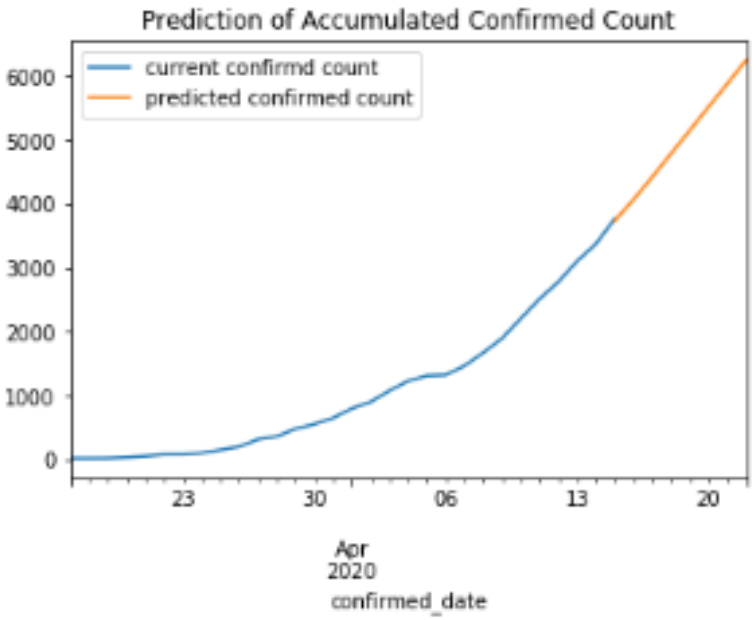


Рисунок 8.14– Графічно представлений прогноз для України на 7 днів

Розглянемо на рисунках 8.15 – 8.22 результати прогнозу на основі методу розрахунку ваг «adam» на прикладі країн – США, Росії, Бельгії, України.

```
Predicted data:
2020-04-17    696295
2020-04-18    725327
2020-04-19    754359
2020-04-20    783391
2020-04-21    812422
2020-04-22    841454
2020-04-23    870486
dtype: int32
Real data:
confirmed_date
2020-04-17    699541.0
2020-04-18    732031.0
2020-04-19    758920.0
2020-04-20    784160.0
2020-04-21    811699.0
2020-04-22    840054.0
2020-04-23    869004.0
```

Рисунок 8.15 – Результати прогнозу та існуючих даних для США

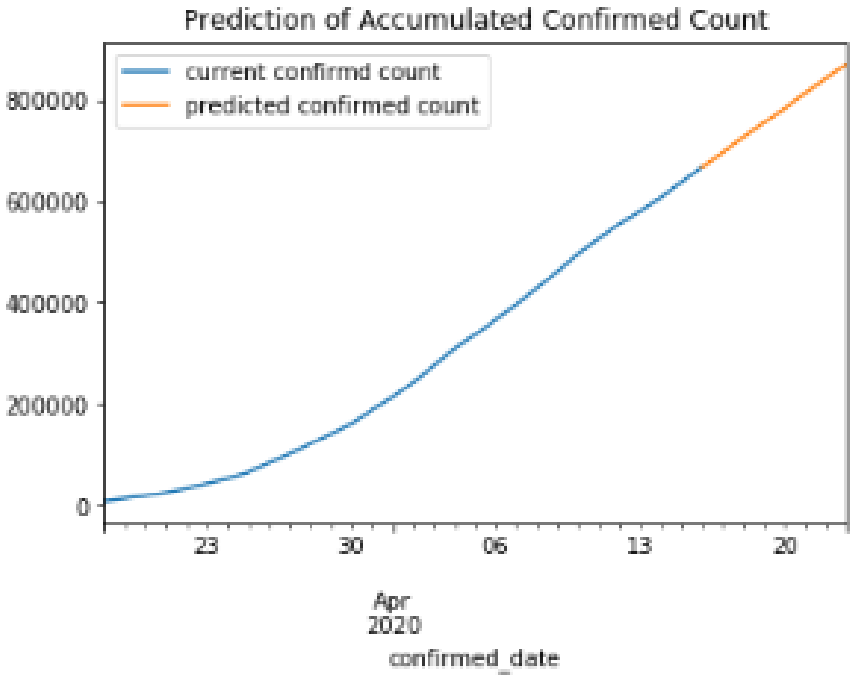


Рисунок 8.16 – Графічно представлений прогноз для США на 7 днів

```
Predicted data:
2020-05-04    142931
2020-05-05    152098
2020-05-06    161265
2020-05-07    170432
2020-05-08    179599
2020-05-09    188766
2020-05-10    197933
dtype: int32
Real data:
confirmed_date
2020-05-04    145268.0
2020-05-05    155370.0
2020-05-06    165929.0
2020-05-07    177160.0
2020-05-08    187859.0
2020-05-09    198676.0
2020-05-10    209688.0
```

Рисунок 8.17 – Результати прогнозу та існуючих даних для Росії

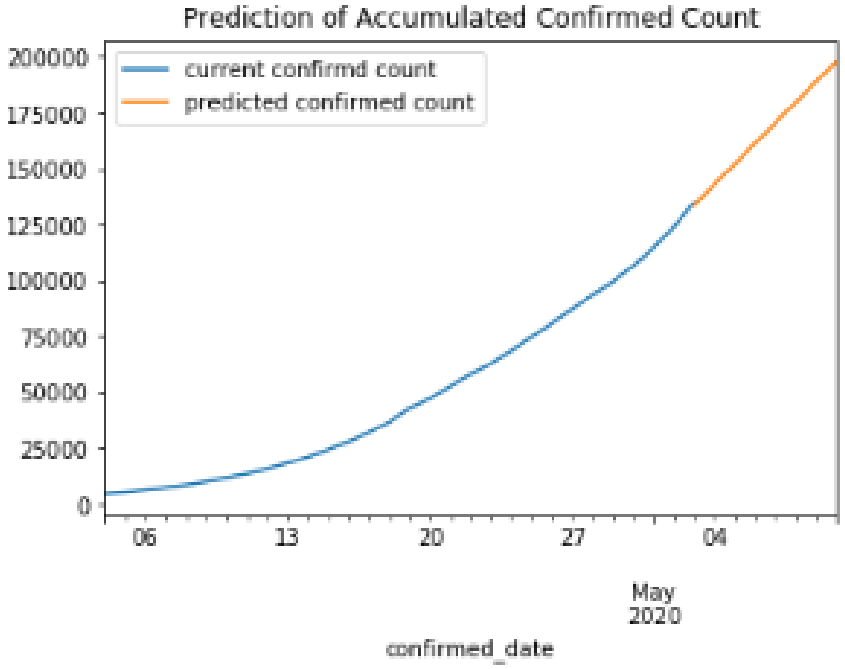


Рисунок 8.18 – Графічно представлений прогноз для Росії на 7 днів

```
Predicted data:
2020-04-08    23553
2020-04-09    24878
2020-04-10    26203
2020-04-11    27528
2020-04-12    28853
2020-04-13    30179
2020-04-14    31504
dtype: int32
Real data:
confirmed_date
2020-04-08    23403.0
2020-04-09    24983.0
2020-04-10    26667.0
2020-04-11    28018.0
2020-04-12    29647.0
2020-04-13    30589.0
2020-04-14    31119.0
```

Рисунок 8.19 – Результати прогнозу та існуючих даних для Бельгії

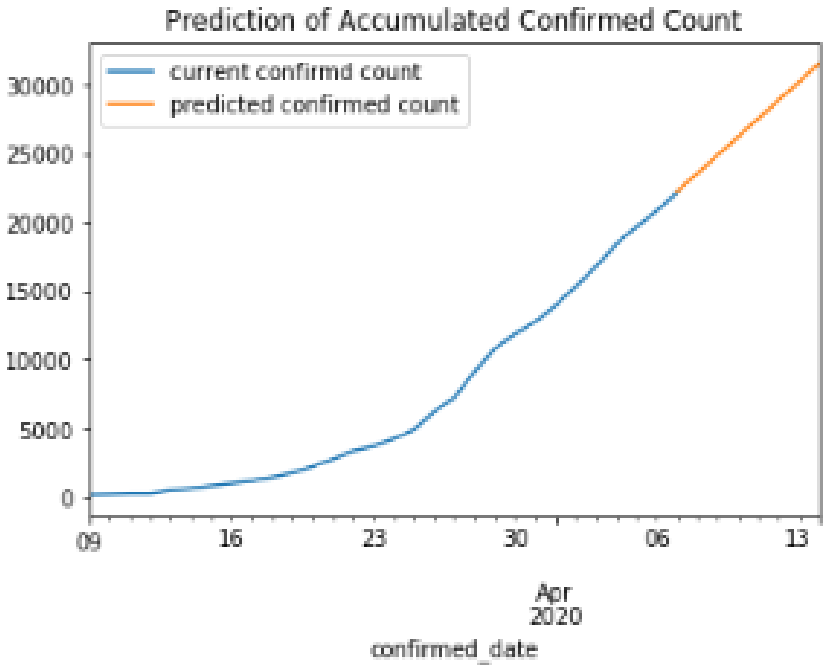


Рисунок 8.20 – Графічно представлений прогноз для Бельгії на 7 днів

```
Predicted data:
2020-04-16    4036
2020-04-17    4344
2020-04-18    4652
2020-04-19    4960
2020-04-20    5268
2020-04-21    5576
2020-04-22    5884
dtype: int32
Real data:
confirmed_date
2020-04-16    4161.0
2020-04-17    4662.0
2020-04-18    5106.0
2020-04-19    5449.0
2020-04-20    5710.0
2020-04-21    6125.0
2020-04-22    6592.0
```

Рисунок 8.21 – Результати прогнозу та існуючих даних для України

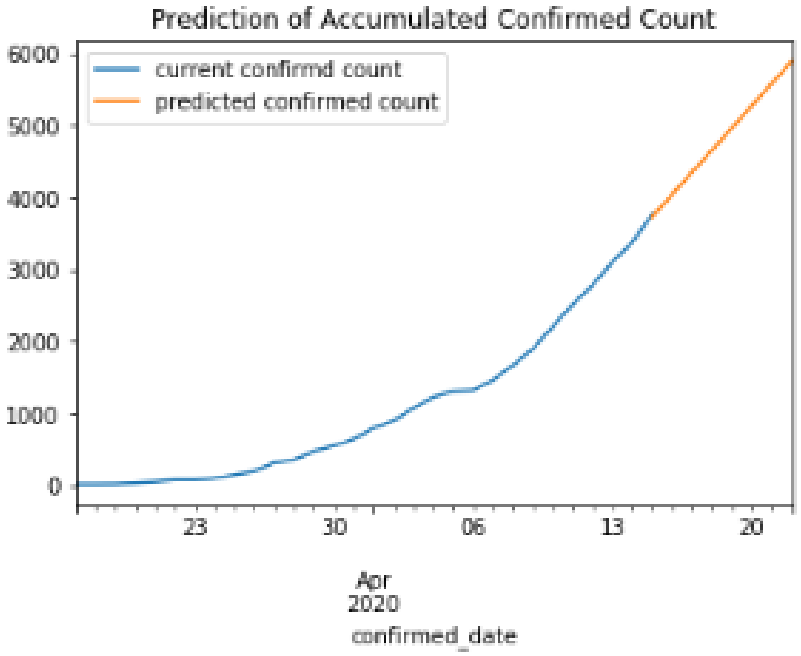


Рисунок 8.22 – Графічно представлений прогноз для України на 7 днів

Порівнюючи результати прогнозів за методами розрахунку ваг «adam» і «lbfgs» (в таблиці 8.1) на основі багатошарового персептрона, можна зробити висновок, що більш точний прогноз був розрахований саме завдяки методу «lbfgs».

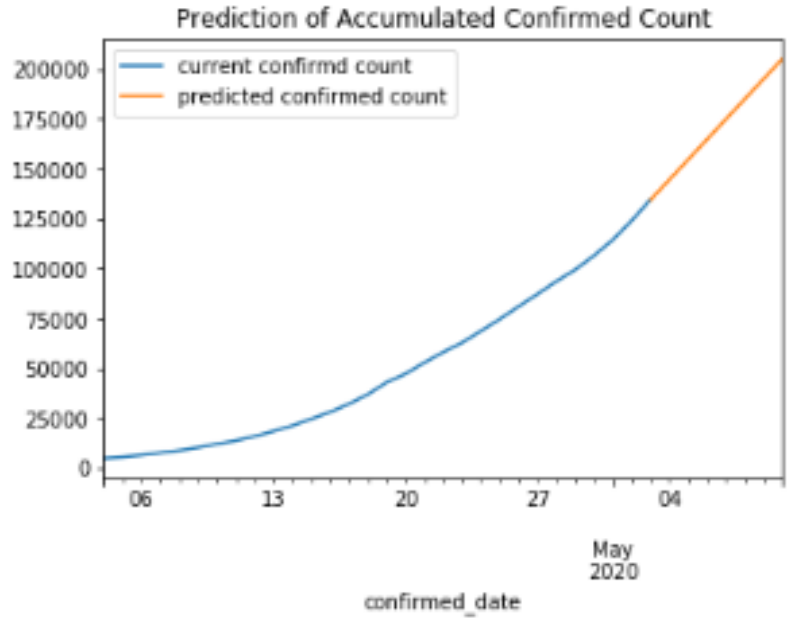
Таблиця 8.1 – Результати за коефіцієнтом детермінації

«adam»	**«lbfgs»**	**Країна**
0.9965443015930223	0.9996449575799649	США
0.8351846232961018	0.9996449575799649	Росія
0.9708021138931431	0.9791727183603849	Бельгія
0.81268741115100116	0.9407489859285618	Україна

Дослідження впливу різних значень параметру регуляризації alpha (0.00001, 0.1, 0.0005) на результат (рис. 8.23 – 8.53). Результати прогнозу на основі методу «lbfgs» розглянемо на прикладі: США, Росії, Бельгії, України, Мексики.

```
Predicted data:
2020-05-04    144639
2020-05-05    154762
2020-05-06    164885
2020-05-07    175009
2020-05-08    185132
2020-05-09    195255
2020-05-10    205378
dtype: int32
Real data:
confirmed_date
2020-05-04    145268.0
2020-05-05    155370.0
2020-05-06    165929.0
2020-05-07    177160.0
2020-05-08    187859.0
2020-05-09    198676.0
2020-05-10    209688.0
```

Рисунок 8.23 – Результати прогнозу та існуючих даних для Росії при α=0.00001

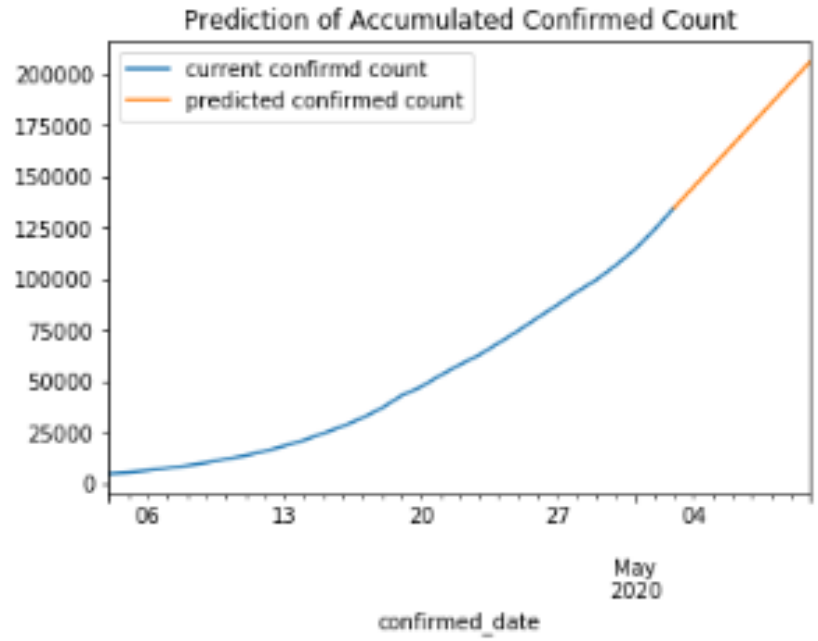



```
0.9845969451656252
```

Рисунок 8.24 – Графічно представлений прогноз для Росії на 7 днів при α=0.00001

```
Predicted data:
2020-05-04    144853
2020-05-05    155087
2020-05-06    165322
2020-05-07    175556
2020-05-08    185791
2020-05-09    196025
2020-05-10    206260
dtype: int32
Real data:
confirmed_date
2020-05-04    145268.0
2020-05-05    155370.0
2020-05-06    165929.0
2020-05-07    177160.0
2020-05-08    187859.0
2020-05-09    198676.0
2020-05-10    209688.0
```

Рисунок 8.25 – Результати прогнозу та існуючих даних для Росії при α=0.1

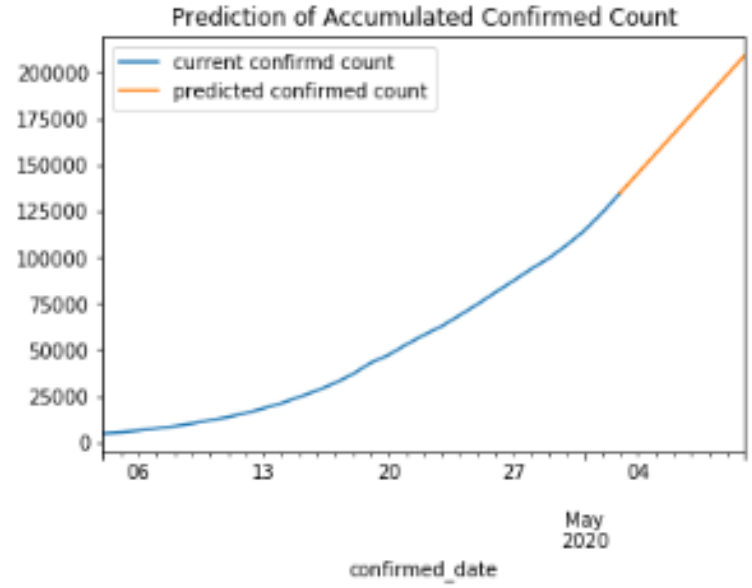


0.9910499681404762

Рисунок 8.26 – Графічно представлений прогноз для Росії на 7 днів при α=0.1

```
Predicted data:
2020-05-04    145331
2020-05-05    155971
2020-05-06    166611
2020-05-07    177252
2020-05-08    187892
2020-05-09    198533
2020-05-10    209173
dtype: int32
Real data:
confirmed_date
2020-05-04    145268.0
2020-05-05    155370.0
2020-05-06    165929.0
2020-05-07    177160.0
2020-05-08    187859.0
2020-05-09    198676.0
2020-05-10    209688.0
```

Рисунок 8.27 – Результати прогнозу та існуючих даних для Росії при α=0.0005

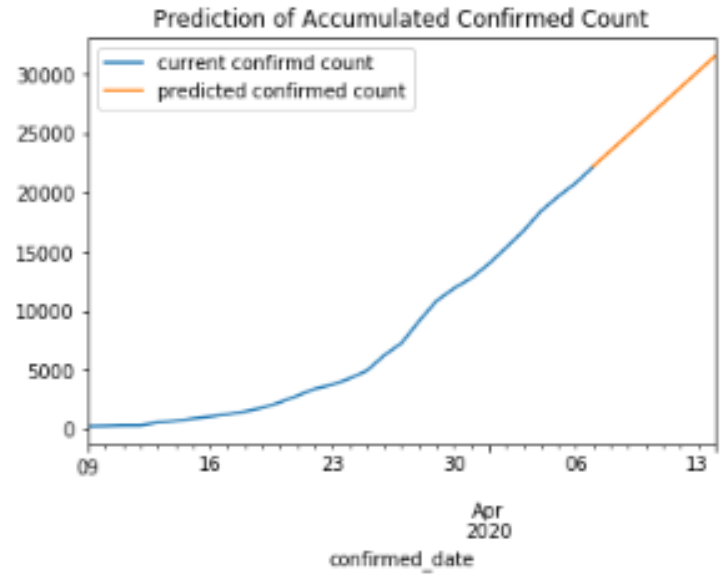



```
0.9996449575799649
```

Рисунок 8.28 – Графічно представлений прогноз для Росії на 7 днів при α=0.0005

```
Predicted data:
2020-04-08    23551
2020-04-09    24881
2020-04-10    26211
2020-04-11    27541
2020-04-12    28879
2020-04-13    30218
2020-04-14    31556
dtype: int32
Real data:
confirmed_date
2020-04-08    23403.0
2020-04-09    24983.0
2020-04-10    26667.0
2020-04-11    28018.0
2020-04-12    29647.0
2020-04-13    30589.0
2020-04-14    31119.0
```

Рисунок 8.29 – Результати прогнозу та існуючих даних для Бельгії при α=0.00001




```
0.9721874387821631
```

Рисунок 8.30 – Графічно представлений прогноз для Бельгії на 7 днів при α=0.00001

```
Predicted data:
2020-05-04    144853
2020-05-05    155087
2020-05-06    165322
2020-05-07    175556
2020-05-08    185791
2020-05-09    196025
2020-05-10    206260
dtype: int32
Real data:
confirmed_date
2020-05-04    145268.0
2020-05-05    155370.0
2020-05-06    165929.0
2020-05-07    177160.0
2020-05-08    187859.0
2020-05-09    198676.0
2020-05-10    209688.0
```

Рисунок 8.31 – Результати прогнозу та існуючих даних для Бельгії на 7 днів при α=0.1

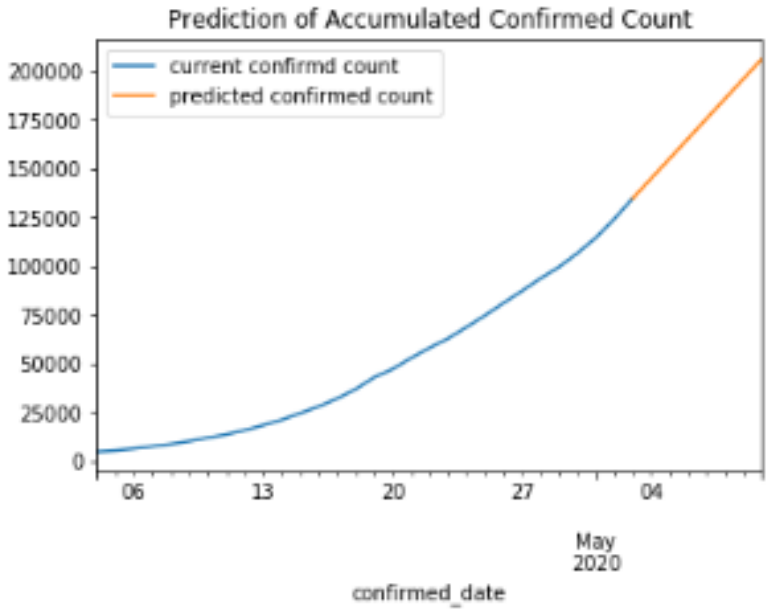
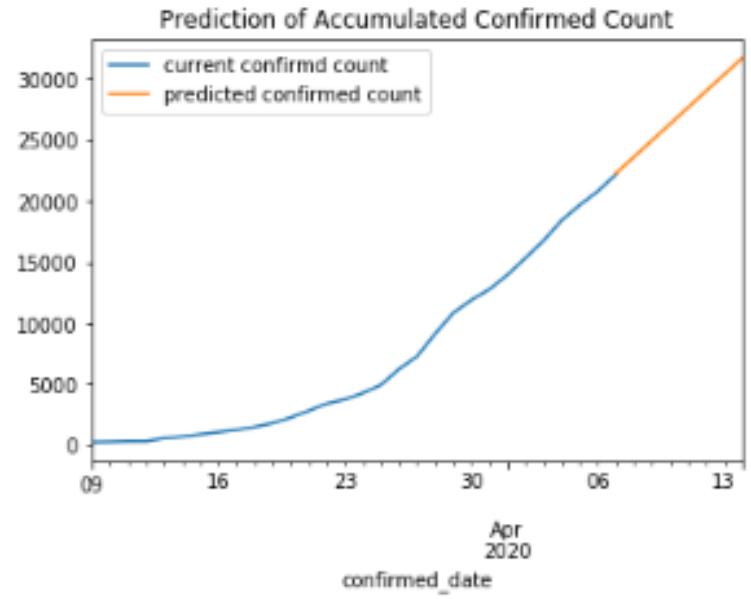


Рисунок 8.32 – Графічно представлений прогноз для Бельгії на 7 днів при α=0.1

```
Predicted data:
2020-04-08    23630
2020-04-09    24983
2020-04-10    26336
2020-04-11    27689
2020-04-12    29042
2020-04-13    30395
2020-04-14    31747
dtype: int32
Real data:
confirmed_date
2020-04-08    23403.0
2020-04-09    24983.0
2020-04-10    26667.0
2020-04-11    28018.0
2020-04-12    29647.0
2020-04-13    30589.0
2020-04-14    31119.0
```

Рисунок 8.33 – Результати прогнозу та існуючих даних для Бельгії на 7 днів при α=0.0005




```
0.9791727183603849
```

Рисунок 8.34 – Графічно представлений прогноз для Бельгії на 7 днів при α=0.0005

```
Predicted data:
2020-04-16    4065
2020-04-17    4420
2020-04-18    4785
2020-04-19    5151
2020-04-20    5518
2020-04-21    5884
2020-04-22    6250
dtype: int32
Real data:
confirmed_date
2020-04-16    4161.0
2020-04-17    4662.0
2020-04-18    5106.0
2020-04-19    5449.0
2020-04-20    5710.0
2020-04-21    6125.0
2020-04-22    6592.0
```

Рисунок 8.35 – Результати прогнозу та існуючих даних для України при α=0.00001

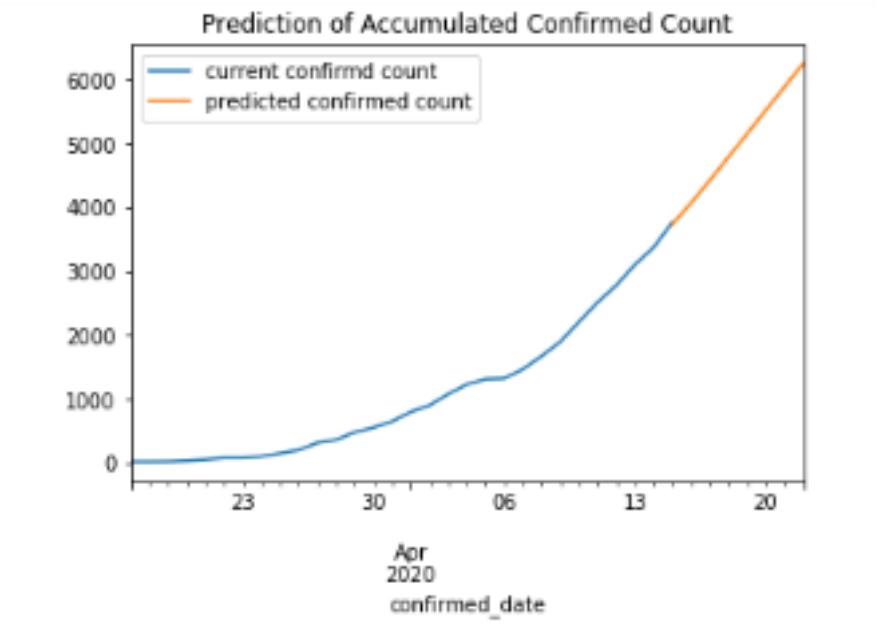
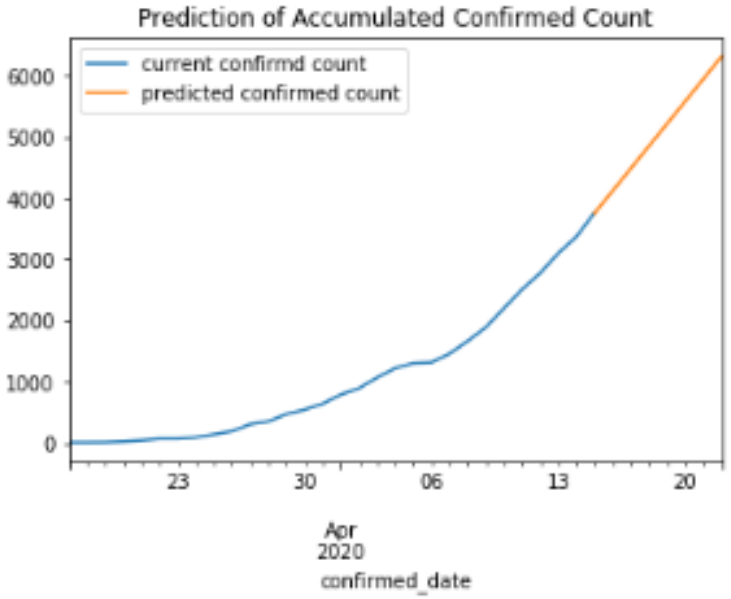


Рисунок 8.36 – Графічно представлений прогноз для України на 7 днів при α=0.00001

```
Predicted data:
2020-04-16    4120
2020-04-17    4485
2020-04-18    4850
2020-04-19    5215
2020-04-20    5580
2020-04-21    5946
2020-04-22    6311
dtype: int32
Real data:
confirmed_date
2020-04-16    4161.0
2020-04-17    4662.0
2020-04-18    5106.0
2020-04-19    5449.0
2020-04-20    5710.0
2020-04-21    6125.0
2020-04-22    6592.0
```

Рисунок 8.37 – Результати прогнозу та існуючих даних для України на 7 днів при α=0.1

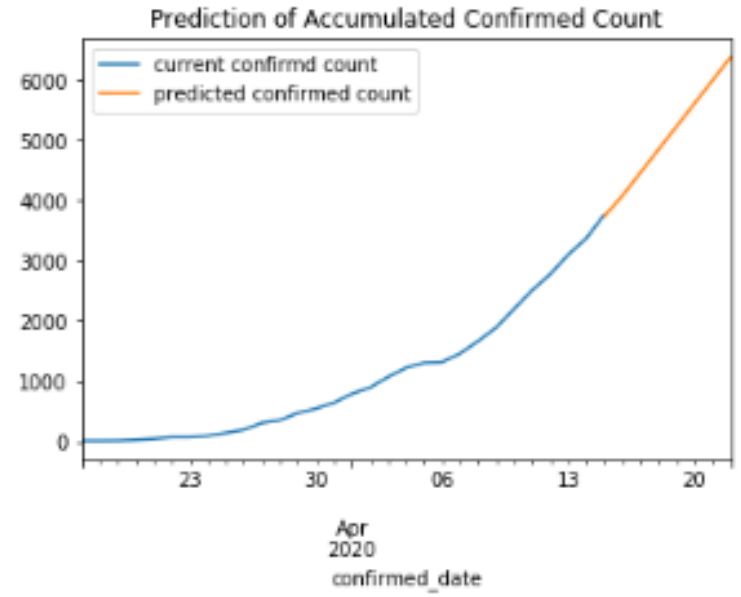



```
0.9246899684210446
```

Рисунок 8.38 – Графічно представлений прогноз для України на 7 днів при α=0.1

```
Predicted data:
2020-04-16    4077
2020-04-17    4460
2020-04-18    4843
2020-04-19    5225
2020-04-20    5608
2020-04-21    5991
2020-04-22    6374
dtype: int32
Real data:
confirmed_date
2020-04-16    4161.0
2020-04-17    4662.0
2020-04-18    5106.0
2020-04-19    5449.0
2020-04-20    5710.0
2020-04-21    6125.0
2020-04-22    6592.0
```

Рисунок 8.39 – Результати прогнозу та існуючих даних для України на 7 днів при α=0.0005




```
0.9407489859285618
```

Рисунок 8.40 – Графічно представлений прогноз для України на 7 днів при α=0.0005

```
Predicted data:
2020-04-17    702301
2020-04-18    732720
2020-04-19    763139
2020-04-20    793558
2020-04-21    823976
2020-04-22    854395
2020-04-23    884814
dtype: int32
Real data:
confirmed_date
2020-04-17    699541.0
2020-04-18    732031.0
2020-04-19    758920.0
2020-04-20    784160.0
2020-04-21    811699.0
2020-04-22    840054.0
2020-04-23    869004.0
```

Рисунок 8.41 – Результати прогнозу та існуючих даних для США при α=0.00001

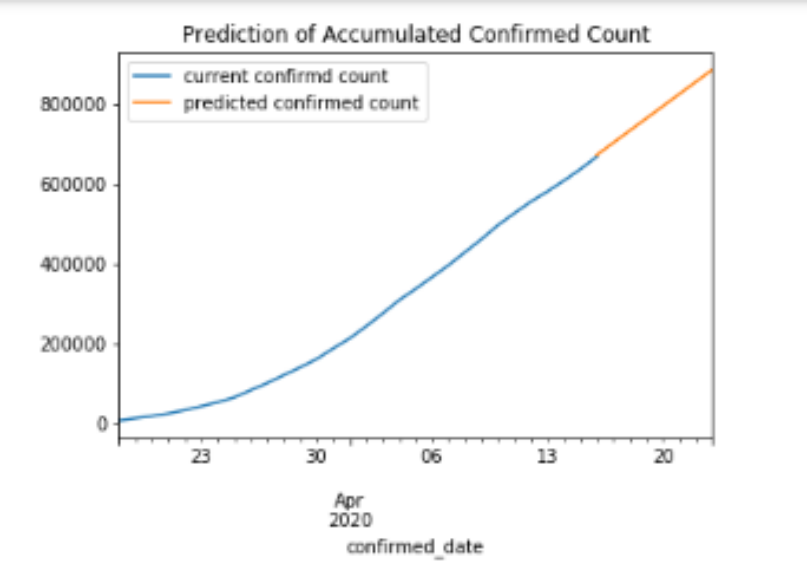


Рисунок 8.42 – Графічно представлений прогноз для США на 7 днів при α=0.00001

```
Predicted data:
2020-04-17    702241
2020-04-18    732651
2020-04-19    763062
2020-04-20    793473
2020-04-21    823884
2020-04-22    854295
2020-04-23    884706
dtype: int32
Real data:
confirmed_date
2020-04-17    699541.0
2020-04-18    732031.0
2020-04-19    758920.0
2020-04-20    784160.0
2020-04-21    811699.0
2020-04-22    840054.0
2020-04-23    869004.0
```

Рисунок 8.43 – Результати прогнозу та існуючих даних для США при α=0.1

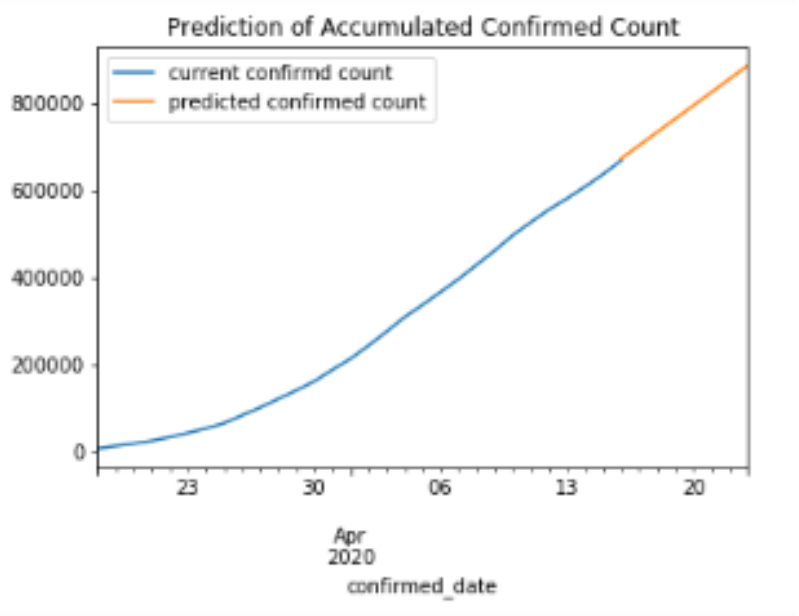


Рисунок 8.44 – Графічно представлений прогноз для США на 7 днів при α=0.1

```
Predicted data:
2020-04-17    702257
2020-04-18    732668
2020-04-19    763080
2020-04-20    793491
2020-04-21    823903
2020-04-22    854314
2020-04-23    884726
dtype: int32
Real data:
confirmed_date
2020-04-17    699541.0
2020-04-18    732031.0
2020-04-19    758920.0
2020-04-20    784160.0
2020-04-21    811699.0
2020-04-22    840054.0
2020-04-23    869004.0
```

Рисунок 8.45 – Результати прогнозу та існуючих даних для США при α=0.0005

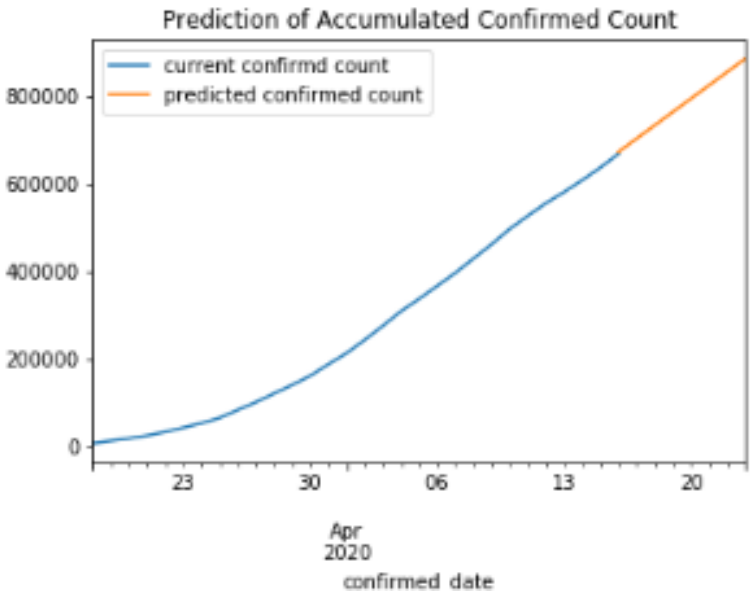


Рисунок 8.46 – Графічно представлений прогноз для США на 7 днів при α=0.0005

```
Predicted data:
2020-05-08    30895
2020-05-09    32428
2020-05-10    33960
2020-05-11    35492
2020-05-12    37025
2020-05-13    38557
2020-05-14    40090
dtype: int32
Real data:
confirmed_date
2020-05-08    31522.0
2020-05-09    33460.0
2020-05-10    35022.0
2020-05-11    36327.0
2020-05-12    38324.0
2020-05-13    40186.0
2020-05-14    42595.0
```

Рисунок 8.47 – Результати прогнозу та існуючих даних для Мексики при α=0.00001

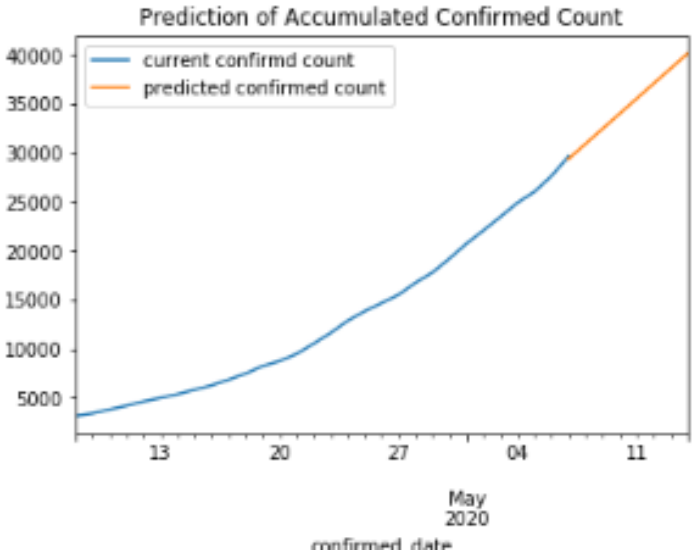


Рисунок 8.48 – Графічно представлений прогноз для Мексики на 7 днів при α=0.00001

```
Predicted data:
2020-05-08    30666
2020-05-09    32100
2020-05-10    33534
2020-05-11    34968
2020-05-12    36402
2020-05-13    37836
2020-05-14    39270
dtype: int32
Real data:
confirmed_date
2020-05-08    31522.0
2020-05-09    33460.0
2020-05-10    35022.0
2020-05-11    36327.0
2020-05-12    38324.0
2020-05-13    40186.0
2020-05-14    42595.0
```

Рисунок 8.49 – Результати прогнозу та існуючих даних для Мексики при α=0.1

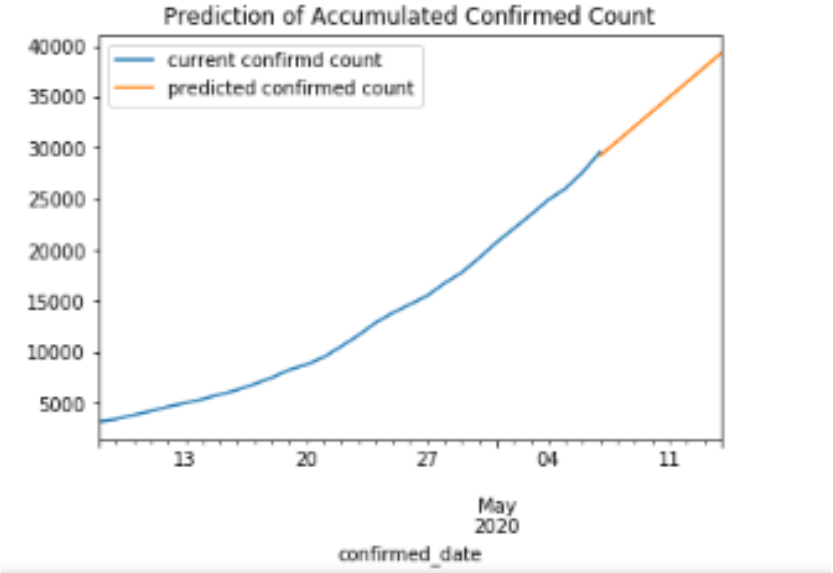


Рисунок 8.50 – Графічно представлений прогноз для Мексики на 7 днів при α=0.1

```
Predicted data:
2020-05-08    31025
2020-05-09    32632
2020-05-10    34239
2020-05-11    35846
2020-05-12    37453
2020-05-13    39061
2020-05-14    40668
dtype: int32
Real data:
confirmed_date
2020-05-08    31522.0
2020-05-09    33460.0
2020-05-10    35022.0
2020-05-11    36327.0
2020-05-12    38324.0
2020-05-13    40186.0
2020-05-14    42595.0
```

Рисунок 8.51 – Результати прогнозу та існуючих даних для Мексики при α=0.0005

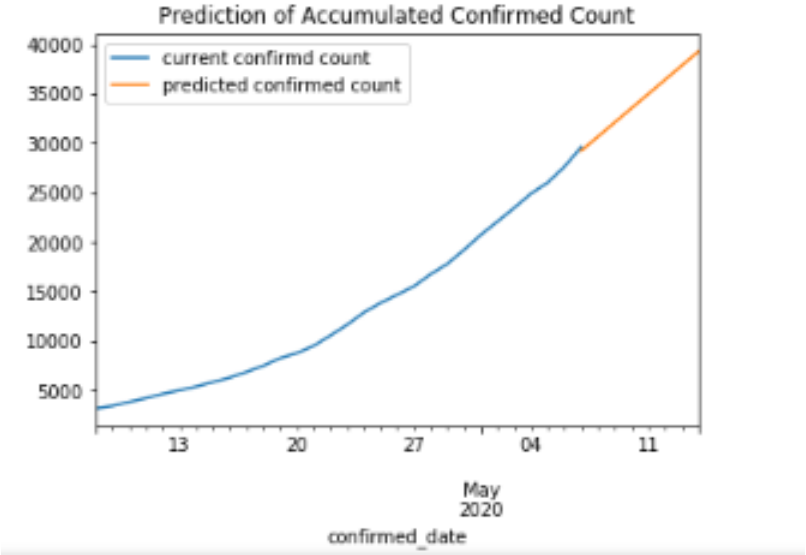


Рисунок 8.52 – Графічно представлений прогноз для Мексики на 7 днів при α=0.0005

Дослідження впливу різних значень параметру регуляризації alpha (0.00001, 0.1, 0.0005) на результат оцінки прогнозу (табл. 8.2).

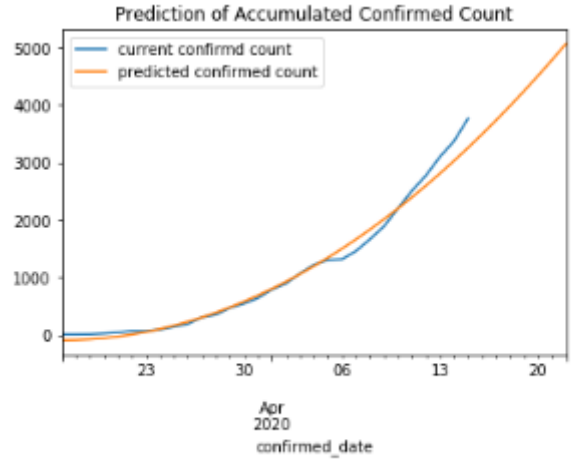
Таблиця 8.2 – Оцінка прогнозу за коефіцієнтом детермінації – R^2

0.00001	**0.1**	**0.0005**	**Країна**
0.972188254380118	0.9726049471137	0.972520064821371	США
0.984596945165625	0.9910499681404	0.999644957579964	Росія
0.970676823456843	0.9742752107468	0.979172718360384	Бельгія
0.70498970266615	0.8734970788957	0.940748985928561	Україна
0.672536994011320	0.7886150474851	0.896098512985805	Мексика

Проаналізувавши таблицю 8.1 – 8.2 можна зробити висновок, що найкращий прогноз був створений на основі багатошарового персептрона для задачі регресії за допомогою методів розрахунку ваг «lbfgs» при коефіцієнті регуляризації α=0.0005 та архітектурі [32, 26, 10,].

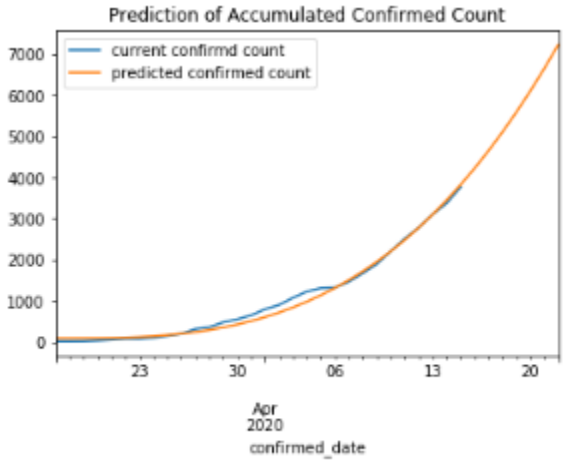
8.3.3 Результати і аналіз моделей на основі методу опорних векторів

Дослідження поліномів різного ступеня по коефіцієнту детермінації на рисунках 8.53 – 8.60 для моделі нелінійної регресії з поліноміальним ядром.



-3.9637707260031485

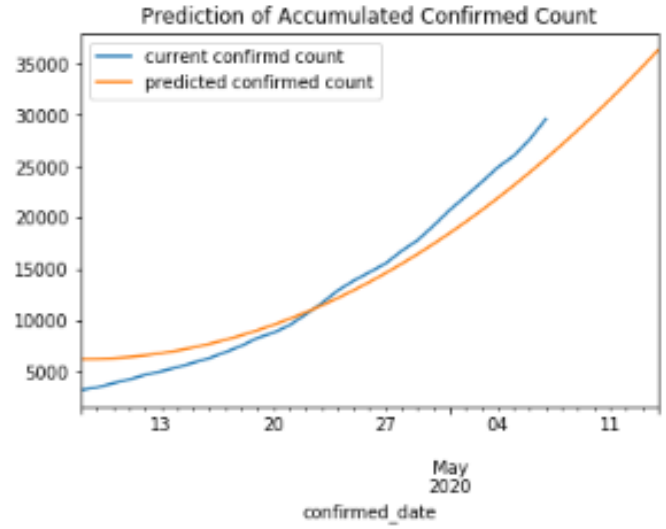
Рисунок 8.53 – Прогноз з поліномом 2-го ступеня та оцінка прогнозу для Бельгії.



0.8769564806439968

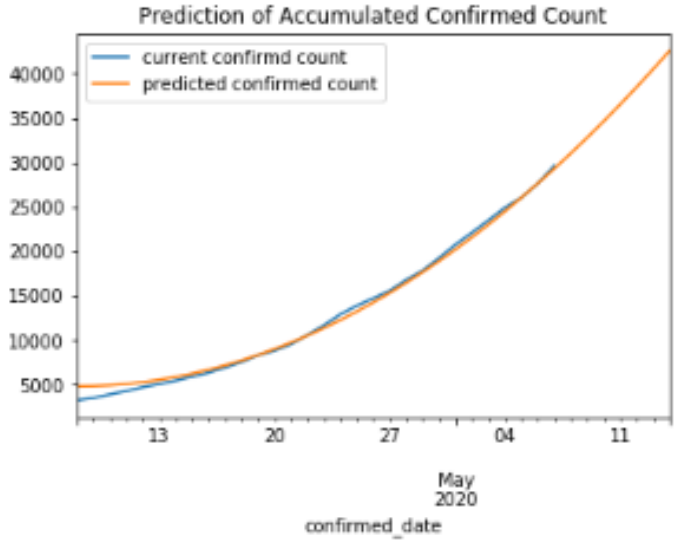
Рисунок 8.54 – Прогноз з поліномом 3-го ступеня та оцінка для Бельгії.

Дослідження комбінації гіперпараметрів: epsilon та C на рисунках 8.55-8..




```
-1.8664338960321594
```

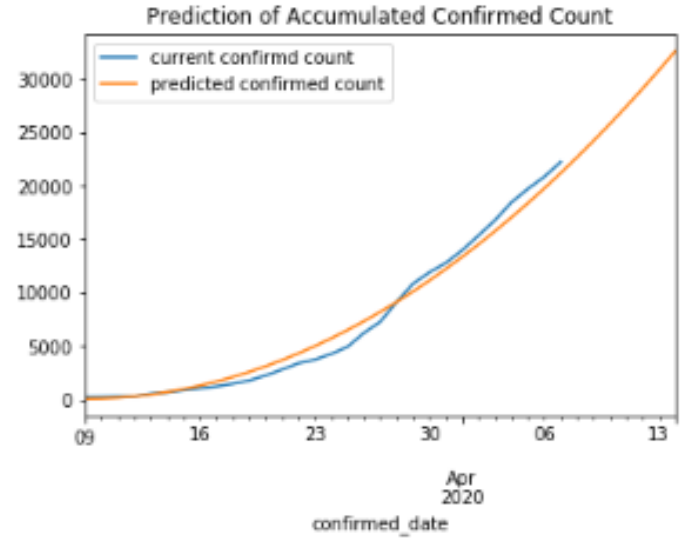
Рисунок 8.55 – Прогноз з epsilon=0.1, C=20 для Мексики




```
0.9902607244669575
```

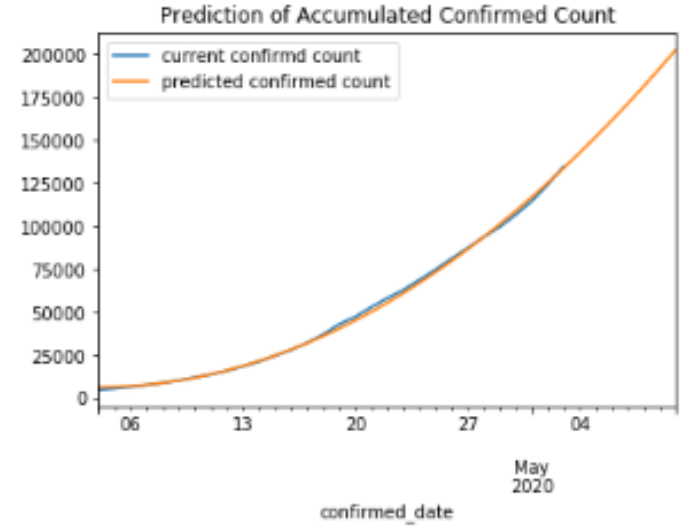
Рисунок 8.56 – Прогноз з epsilon=0.1, C=30 для Мексики

На рисунках 8.57 – 8.60 зображено результат найкращої моделі нелінійної регресії з поліноміальним ядром (поліном 3-го ступеня, epsilon=0.1, C=30) на прикладі країн: Бельгія, Росія, Мексика та Україна.



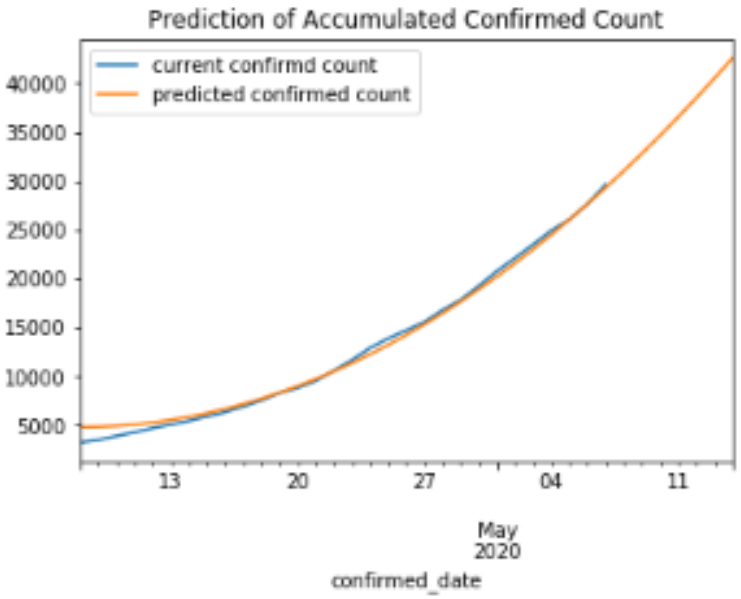

```
0.935031786116376
```

Рисунок 8.57 – Графічно представлений прогноз для Бельгії на 7 днів та оцінка прогнозу



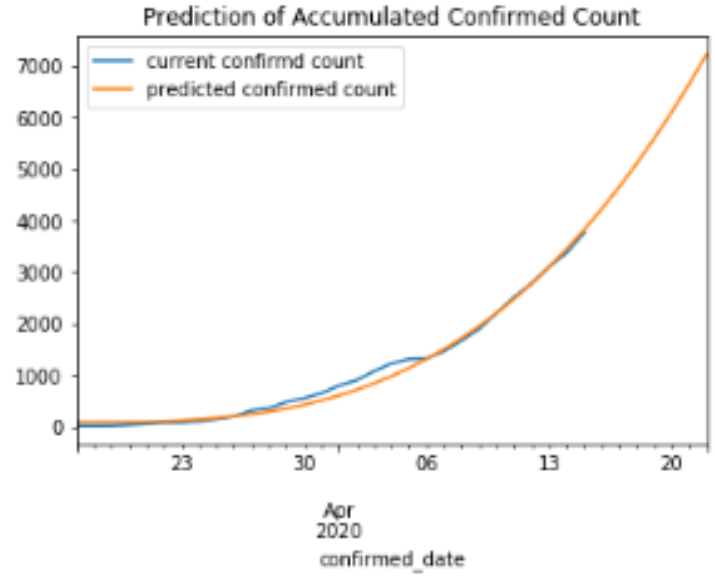

```
0.9259077408231322
```

Рисунок 8.58 – Графічно представлений прогноз для Росії на 7 днів та оцінка прогнозу




```
0.9902607244669575
```

Рисунок 8.59 – Графічно представлений прогноз для Мексики на 7 днів




```
0.8769564806439968
```

Рисунок 8.60 – Графічно представлений прогноз для України на 7 днів

8.3.4 Результати і аналіз на основі ансамблів моделей

Перед тим як досліджувати ансамбль методів треба поділити вибірку часового ряду на тренувальну та тестову (рис. 8.61).

```
def trainModels(df):
    # split into train and test
    idx = train_and_test(df)
    train = idx[0]
    test = idx[1]
    test_1 = test[0]
    test_2 = test[1]
    train_data = df[train[0]:train[1]]
    test_data = df[test_1:test_2]
    x = np.arange(len(train_data)).reshape(-1,1)
    y = train_data['daily_count'].values
    test = np.arange(len(test_data)).reshape(-1, 1)
    y2 = test_data['daily_count'].values
    pick_date = df[(df['daily_count'] == df['daily_count'].max())].index[0]
```

Рисунок 8.61 – Розподіл на тестову та тренувальну вибірку

Далі створення ансамбля моделей на основі методів: Random Forest, Gradient Boosting, Ada Bosting Regressor, Bosting Regressor, Bagging Regressor, Voting Regressor, Stacking Regressor. Та досліджння їх параметрів, щоб отримати в результаті найкращий ансамбль моделей (рис. 8.62).

```
### Random Forest Model
rand_forest_model = RandomForestRegressor(n_estimators=1000, max_features=1, oob_score=True, random_state=115)
rand_forest_model.fit(x,y)
print('R-square value of Random Forest Model: ', rand_forest_model.score(test, y2))

### Gradient Boosting
gr_boosting_model = GradientBoostingRegressor(n_estimators=500, learning_rate=0.1, subsample=1, max_features=1, loss='ls')
gr_boosting_model.fit(x,y)
print('Gradient Boosting:', gr_boosting_model.score(test, y2))

# Ada Boosting Regressor
ad_boost_model = AdaBoostRegressor(base_estimator=None, n_estimators=50, learning_rate=1.0, loss='linear',
                                   random_state=None)
ad_boost_model.fit(x,y)
print('Ada Boosting Regressor:', ad_boost_model.score(test, y2))

# Bagging Regressor
bagging_model = BaggingRegressor(base_estimator=None, n_estimators=10, max_samples=1.0, max_features=1.0,
                                 bootstrap=True, bootstrap_features=False, oob_score=False, warm_start=False, n_jobs=None,
bagging_model.fit(x,y)
print('Bagging Regressor', bagging_model.score(test, y2))

# Voting Regressor
voting_model = VotingRegressor(estimators=100, weights=None, n_jobs=None, verbose=False)
voting_model.fit(x,y)
print('Voting Regressor', voting_model.score(test, y2))

# Stacking Regressor
stacking_model = StackingRegressor(estimators=100, final_estimator=None, cv=None, n_jobs=None,
                                   passthrough=False, verbose=0)
stacking_model.fit(x,y)
print('Stacking Regressor', stacking_model.score(test, y2))
```

Рисунок 8.62 – Створення ансамбля моделей

Перевірка прогнозу моделі ансамбля коефіцієнтом детермінації R^2 на рисунку 8.63.

```
R-square value of Random Forest Model: 0.9713020163118805
R-square value of Gradient Boosting: 0.984212591296279
R-square value of Ada Boosting Regressor: 0.9673130769242105
R-square value of Bagging Regressor: 0.9056376408076269
R-square value of Voting Regressor: 0.7239865312461513
R-square value of Stacking Regressor: 0.88398253176641513
```

Рисунок 8.63 – Результат оцінки якості прогнозу для ансамбля моделей

8.4 Висновки

Проаналізувавши всі моделі та методи, можна зробити висновок, що найвисоку точність прогнозування на 7 днів показала нелінійна модель багатошарового персептрона для задачі регресії. Її середня оцінка прогнозу – 0.957637047935217 на прикладі 5-ти країн.

РОЗДІЛ 9
ФУНКЦІОНАЛЬНОВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

Вступ

У даному розділі проводиться оцінка основних характеристик програмного продукту, призначеного прогнозування поширення коронавірусу та аналізу впливу карантинних мір в різних країнах світу.

Програмний продукт був розроблений за допомогою мови програмування Python у середовищі Jupiter. Результати прогнозування призначені для використання на персональних комп'ютерах під управлінням операційної системи Windows, Linux та Mac OS.

9.1 Постановка задачі технікоекономічного аналізу

Під час застосування методу ФВА для проведення технікоекономічного аналізу розробленого продукту необхідно обрати систему показників якості програмного продукту. Він має відповідати наступним технічним вимогам:

- функціонування на персональних комп'ютерах зі стандартним набором компонент;
- швидкодія та обробка великого об'єму даних у режимі реального часу;
- зручність та простота у роботі для користувача або для розробника іншого програмного забезпечення на основі програмного продукту, що аналізується;
- мінімальні витрати на впровадження програмного продукту.

9.2 Обґрунтування функцій та параметрів програмного продукту

Головна функція F_0 – використання програмного продукту, який аналізує отримує на вхід дані та будує його модель для прогнозування поширення епідемії. Виходячи з конкретної мети, можна виділити наступні основні функції ПП:

F_1 – вибір мови програмування;

F_2 – використання готових бібліотек;

F_3 – середовище розробки.

Кожна з основних функцій може мати декілька варіантів реалізації.

Функція F_1:

а) мова програмування Python;

б) мова програмування C++;

Функція F_2:

а) використання готової бібліотеки;

б) написання алгоритмів роботи з даними вручну.

Функція F_3:

а) середовище розробки Jupiter;

б) середовище розробки PyCharm.

Варіанти реалізації основних функцій наведені у морфологічній карті системи (рис. 9.1).

Спираючись на карту була побудована позитивно-негативна матриця (табл. 9.1).

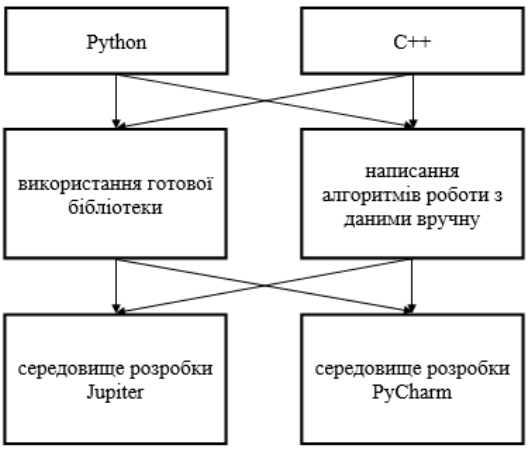


Рисунок 9.1 – Морфологічна карта

Таблиця 9.1 – Позитивно-негативна матриця

Основна функція	**Варіант реалізації**	**Переваги**	**Недоліки**
F1	А	Більше пристосований для роботи з великими даними, кросплатформний	Динамічна типізація
	Б	Відсутня динамічна типізація	Більше часу для написання коду, погано працюю з великими даними
F2	А	Легкість реалізації, економія часу	Не завжди кінцева реалізація
	Б	Оптимально для власних продуктів	Велика кількість помилок
F3	А	Широкий вибір можливостей	Відсутня відладка коду
	Б	Відлатка коду	Додаткова інсталяція

Таким чином, будемо розглядати такі варіанти реалізації ПП:

1. F1а – F2а – F3а;
2. F1а – F2а – F3б

9.3 Обґрунтування системи параметрів ПП

9.3.1 Опис параметрів

На підставі даних про основні функції, що повинен реалізувати програмний продукт, вимог до нього, визначаються основні параметри виробу, що будуть використані для розрахунку коефіцієнта технічного рівня.

Для того, щоб охарактеризувати програмний продукт, будемо використовувати наступні параметри: X1 – швидкодія мови програмування, X2 – об’єм пам’яті для збереження даних, X3– потенційний об’єм програмного/

X1 відображає швидкодію операцій залежно від обраної мови програмування. X2 відображає об’єм пам’яті в оперативній пам’яті персонального комп’ютера, необхідний для збереження та обробки даних під час виконання програми. X3 показує розмір програмного коду, який необхідно створити безпосередньо розробнику.

9.3.2 Кількісна оцінка параметрів

Для характеристики прототипу програмного додатку використовуємо параметри X1 – X3. Визначаємо мінімальні, середні отримуванні та максимально допустимі значення в таблиці 9.2.

Таблиця 9.2 – Система параметрів додатку

Найменування параметру	**Позначення параметру**	**Значення параметру**		
		Гірші	**Середнє**	**Кращі**
Швидкодія мови програмування, с	X1	15	7	2
Об’єм пам’яті для збереження даних, гб	X2	31	13	4
Об’єм програмного коду, строк	X3	3000	1800	900

За даними таблиці 9.2 будуються графічні характеристики параметрів (рис.9.2 – 9.4).

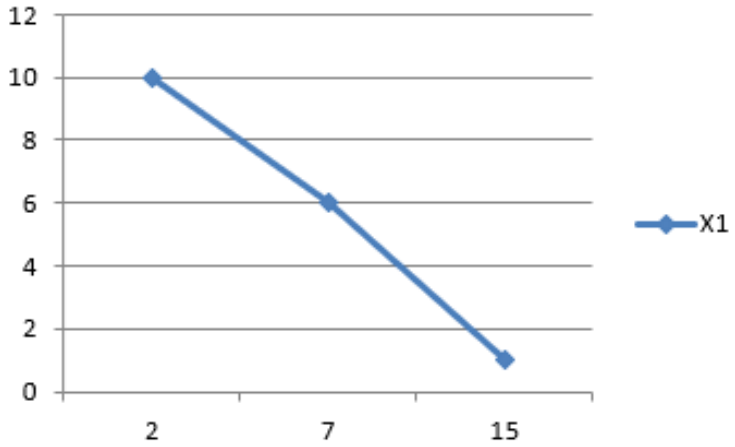


Рисунок 9.2 – Значення параметра X1 – швидкодії мови програмування

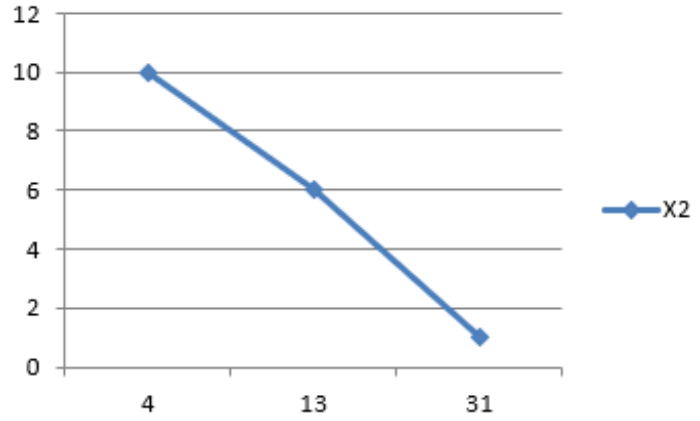


Рисунок 9.3 – Значення параметру X2 – об’єм пам’яті для збереження даних

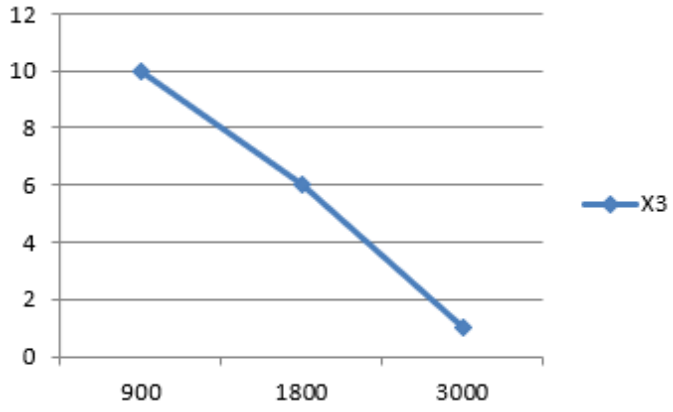

12
10
8
6
4
2
0
900
1800
3000
X3

Рисунок 9.4 – Значення параметру X3 – потенційний об’єм програмного коду

9.3.3 Аналіз експертного оцінювання параметрів

Ранги варіюються від 1 до 4. Результати наведені в таблиці 9.3 – 9.4

Таблиця 9.3 – Результат оцінки параметрів

Параметр	**Ранг параметру по оцінці експерта**							**Сума рангів, R_i**	**ВідхиленняΔ_i**	**Квадрат відхилення, $(\Delta_i)^2$**
	1	**2**	**3**	**4**	**5**	**6**	**7**			
X1	1	2	2	3	2	2	2	14	0	0
X2	2	1	1	1	1	1	1	8	-6	36
X3	3	3	3	2	3	3	3	20	6	36
Разом	6	6	6	6	6	6	6	42	0	72

$$W = \frac{12S}{N^2(n^3 - n)} = \frac{12 * 72}{49 * (27 - 3)} = 0{,}7346 > W_k = 0{,}67 \qquad (9.1)$$

Порахуємо коефіцієнт узгодженості по формулі:

Ранжування можна вважати достовірним, тому що знайдений коефіцієнт узгодженості перевищує нормативний, котрий дорівнює 0,67.

Таблиця 9.4 – Попарне зрівняння параметрів

Параметри	**Експерти**							**Кінцева оцінка**	**Числове значення**
	1	**2**	**3**	**4**	**5**	**6**	**7**		
X1 та X2	<	>	>	>	>	>	>	>	1.5
X1 та X3	<	<	<	>	<	<	<	<	0.5
X2 та X3	<	<	<	<	<	<	<	<	0.5

Як ми бачимо в таблиці 9.5 різниця значень коефіцієнтів вагомості не перевищує 2%, тому більшої кількості ітерацій не потрібно.

Таблиця 9.5 – Розрахунок вагомості параметрів

Параметри	**Параметри xj**			**Перший крок**		**Другий крок**		**Третій крок**	
	X1	**X2**	**X3**	$\mathbf{b_i}$	$\mathbf{К_{ві}}$	$\mathbf{b_i}$	$\mathbf{К_{ві}}$	$\mathbf{b_i}$	$\mathbf{К_{ві}}$
X1	1	1,5	0,5	3	0,333	8	0,320	22	0,318
X2	0,5	1	0,5	2	0,222	5,5	0,220	15,25	0,221
X3	1,5	1,5	1	4	0,444	11,5	0,460	31,75	0,460
Загалом:				9	1	25	1	69	1,00

9.4 Аналіз рівня якості варіантів реалізації функцій

Визначаємо рівень якості кожного варіанту (табл. 9.6) виконання основних функцій окремо. Абсолютні значення параметрів X2 та X1 відповідають технічним вимогам умов функціонування даного ПП.

Таблиця 9.6 – рівень якості

Основна функція	**Варіант реалізації**	**Абсолютне значення параметру**	**Бальна оцінка параметру –**	**Коефіцієнт вагомості параметру**	**Коефіцієнт якості**
F1	а)X1	8	5,5	0,318	1,749
F2	а)X2	18	4,2	0,221	0,9383
F3	а) X3	1000	9,5	0,460	4,37
	б) X3	1500	7,33	0,460	3,37

Обрахуємо коефіцієнти якості кожного з варіантів розробки:

$$К_{я1} = 1{,}75 + 0{,}94 + 4{,}37 = 7{,}06$$

$$К_{я2} = 1{,}75 + 0{,}94 + 3{,}37 = 6{,}06$$

Як видно з розрахунків, кращим є перший варіант, для якого коефіцієнт технічного рівня має найбільше значення.

9.5 Економічний аналіз варіантів розробки ПП

Для визначення вартості розробки ПП спочатку проведемо розрахунок трудомісткості.

Всі варіанти включають в себе два окремих завдання:

1. Розробка проекту програмного продукту;
2. Розробка програмної оболонки;

При цьому два варіанти мають різні додаткові завдання:

3. Реалізація методів аналізу;
4. Обробка готового інтрефейсу бібліотек.

Для першого завдання трудомісткість дорівнює: ТР =90 людино-днів. Поправочний коефіцієнт, який враховує вид нормативно-довідкової інформації для першого завдання: КП = 1.7. Поправочний коефіцієнт, який враховує складність контролю вхідної та вихідної інформації для всіх семи завдань рівний 1: КСК = 1. Оскільки при розробці першого завдання використовуються стандартні модулі, врахуємо це за допомогою коефіцієнта КСТ = 0.8. Тоді, загальна трудомісткість програмування першого завдання дорівнює:

$$Т1 = 90 \cdot 1.7 \cdot 0.8 = 122.4 \text{ людино} - \text{днів}$$

Для другого завдання:

$$ТР = 27 \text{ людино} - \text{днів}, КП = 0.9, КСК = 1,$$

$$КСТ = 0.8: Т2 = 27 \cdot 0.9 \cdot 0.8 = 19.44 \text{ людино} - \text{днів}.$$

Для третього завдання (використовується алгоритм другої групи складності, степінь новизни Г з використанням перемінної інформації), тобто ТР =6 людино-днів, $КСК = 1, КСТ = 0.8$:

$$Т2 = 6 \cdot 0.8 = 4{,}8 \text{ людино} - \text{днів}$$

Для четвертого завдання (використовується алгоритм третьої групи складності, степінь новизни Г), тобто ТР =8 людино-днів, КСК = 1, КСТ =0.9:

$$Т2 = 8 \cdot 0.8 = 7{,}2 \text{ людино} - \text{днів}$$

Складаємо трудомісткість відповідних завдань для кожного з обраних варіантів реалізації програми, щоб отримати їх трудомісткість:

$$ТІ = (122.4 + 19.44 + 4.8) \cdot 8 = 1173{,}12 \text{ людино} - \text{годин.}$$

$$ТІІ = (122.4 + 19.44 + 7.2) \cdot 8 = 1186 \text{ людино} - \text{годин.}$$

Найбільш високу трудомісткість має варіант II.

В розробці беруть участь один програміст з окладом 10000 грн., один дата сайнтист з окладом 14000грн. Визначимо зарплату за годину:

$$С_ч = \frac{10000 + 14000}{2 * 8 * 21} = 71{,}42 \text{грн}$$

Зарплата розробників за варіантами становить:

$$СЗП = 71{,}42 \cdot 1173{,}12 \cdot 1.2 = 100541.08 \text{ грн}$$

$$СЗП = 71.42 \cdot 1186 \cdot 1.2 = 101644.9 \text{ грн}$$

Відрахування на соціальний внесок становить 22%:

$$СВІД = СЗП \cdot 0.22 = 100541.08 \cdot 0.22 = 22119.04 \text{ грн}$$

$$СВІД = СЗП \cdot 0.22 = 101644.9 \cdot 0.22 = 22361.88 \text{ грн}$$

Так як одна ЕОМ обслуговує одного програміста з окладом 10000 грн., з коефіцієнтом зайнятості 0,2 то для однієї машини отримаємо:

$$СГ = 12 \cdot М \cdot КЗ = 12 \cdot 10000 \cdot 0{,}2 = 24000 \text{ грн}$$

З урахуванням додаткової заробітної плати:

$$СЗП = СГ \cdot (1 + КЗ) = 24000 \cdot (1 + 0.2) = 28800 \text{ грн}$$

Відрахування на соціальний внесок:

$$СВІД = СЗП \cdot 0.22 = 28800 \cdot 0.22 = 6336 \text{ грн}$$

Амортизаційні відрахування розраховуємо при амортизації 25% та вартості ЕОМ – 24000 грн.

$$СА = КТМ \cdot КА \cdot ЦПР = 1.15 \cdot 0.25 \cdot 24000 = 6900 \text{ грн.}$$

Витрати на ремонт та профілактику розраховуємо як:

$$СР = КТМ \cdot ЦПР \cdot КР = 1.15 \cdot 24000 \cdot 0.05 = 1380 \text{ грн,}$$

де КР– відсоток витрат на поточні ремонти.

Ефективний годинний фонд часу ПК за рік розраховуємо за формулою:

$$\text{ТЕФ} = (\text{ДК} - \text{ДВ} - \text{ДС} - \text{ДР}) \cdot t\text{З} \cdot \text{КВ} = (365 - 104 - 8 - 16) \cdot 8 \cdot 0.9 = 1706.4,$$

Витрати на оплату електроенергії розраховуємо за формулою:

$\text{СЕЛ} = \text{ТЕФ} \cdot N\text{С} \cdot K\text{З} \cdot \text{ЦЕН} = 1706{,}4 \cdot 0{,}7 \cdot 0{,}2 \cdot 1{,}75 = 418{,}07$ грн,

де NС - середньо-споживча потужність приладу; КЗ - коефіцієнт зайнятості приладу; ЦЕН - тариф за 1 КВт-годин електроенергії.

Накладні витрати розраховуємо за формулою:

СН = ЦПР·0.67 = 24000· 0,67 =16080 грн

СЕКС = 28800 + 6336 + 6900 + 1380 + 2090,34 + 16080 = 61524.07 грн.

Собівартість однієї машино-години ЕОМ дорівнюватиме:

СМ-Г = СЕКС/ ТЕФ = 61524.07 /1706.4 = 36,05 грн/час.

Оскільки в даному випадку всі роботи, які пов‘язані з розробкою програмного продукту ведуться на ЕОМ, витрати на оплату машинного часу, в залежності від обраного варіанта реалізації, складає:

СМ = 36,05 · 1328,64 = 47903.97 грн, СМ = 36,05 · 1345,52 = 48512.58 грн

Накладні витрати складають 67% від заробітної плати:

СН = СЗП · 0,67

СН = 100541.08 · 0,67 = 67362,52 грн,

СН = 101644.9 · 0,67 = 68102,08 грн.

Отже, вартість розробки ПП за варіантами становить:

СПП = СЗП+ СВІД+ СМ +СН.

СПП = 100541.08 + 22119.04 + 47903.97 + 67362,52 = 237226.61 грн,

СПП = 101644.9 + 22361.88 + 48512.58 + 68102,08 = 240621.44 грн

9.6 Вибір кращого варіанта ПП техніко-економічного рівня

Розрахуємо коефіцієнт техніко-економічного рівня:

КТЕР1 = 7,06/ 237226.61 = $0,298 \cdot 10^{-4}$

КТЕР2=6,06/240621.44=$0,252 \cdot 10^{-4}$

Як бачимо, найбільш ефективним є перший варіант реалізації програми з коефіцієнтом техніко-економічного рівня КТЕР1= $0,298 \cdot 10^{-4}$.

9.7 Висновки

Після виконання функціонально-вартісного аналізу програмного комплексу, що розроблюється, можна зробити висновок, що оптимальним є перший варіант реалізації програмного продукту. У нього виявився найкращий показник техніко-економічного рівня якості КТЕР = $0,298 \cdot 10^{-4}$. Цей варіант реалізації програмного продукту має такі параметри:

- мова програмування – Python;
- використання бібліотек;
- середовище розробки Jupyter.

ВИСНОВКИ

В роботі виконано порівняльний аналіз методів аналізу часових рядів, таких як модель поліноміальної регресії, метод опорних векторів, моделі багатошарових нейронних мереж прямого розповсюдження, ансамблі моделей.

Побудовано і досліджено моделі нелінійних часових рядів для прогнозування поширення коронавірусу в різних країнах світу. На основі вибраної найкращої моделі побудовано прогноз поширення коронавірусу. Реалізовано програмний продукт з використанням сучасних бібліотек «Scikit-Learn» та «Pandas» для побудови і аналізу моделей машинного навчання, та «Matplotlib» для візуалізації даних.

Отримано, що найвисоку точність прогнозування на 7 днів показала нелінійна модель багатошарового персептрона для задачі регресії. Її середня оцінка прогнозу – 0.957637047935217 на прикладі 5-ти країн.

ПЕРЕЛІК ПОСИЛАНЬ

1. Джефф А., Боротьба з короновірусом, 14.03.2020 URL: https://www.sas.com/ru_ru/insights/articles/analytics/fighting-coronavirus--4-ways-analytics-is-making-a-difference.html

2. Хотеева Ю., Актуальный список стран, в которых введен карантин по коронавирусу, 13.05.2020 URL: https://www.biletik.aero/handbook/blog/polnyy-spisok-stran-v-kotorykh-vveden-karantin-po-koronavirusu-obnovlyaetsya/ (Last accessed: 18.05.2020).

3. Siddiqui U., Coronavirus testing methods: What you need to know, 7.04.2020 URL: https://www.aljazeera.com/news/2020/03/coronavirus-testing-methods-200330142718434.html (Last accessed: 18.05.2020).

4. Joe Hasell, Esteban Ortiz-Ospina, Edouard Mathieu, Hannah Ritchie, Diana Beltekian, Bobbie Macdonald, Max Roser Coronovirus Testing 16.04.2020 URL: https://ourworldindata.org/coronavirus-testing (Last accessed: 18.05.2020).

5. Klaarens S. COVID-19 pandemic Humanity needs leadership and solidarity to defeat the coronavirus 13.03.20 URL: https://www.undp.org/content/undp/en/home/coronavirus.html (Last accessed: 18.05.2020).

6. Richard Pérez-Peña, Peaks, Testing, Lockdowns: How Coronavirus Vocabulary Causes Confusion 03.04.2020 URL: https://www.nytimes.com/2020/04/03/world/europe/coronavirus-language-confusion.html (Last accessed: 18.05.2020).

7. Tom Duszynski, What does ‘recovered from coronavirus’ mean? 4 questions answered about how some survive and what happens next 17.04.2020 URL: https://theconversation.com/what-does-recovered-from-coronavirus-mean-4-questions-answered-about-how-some-survive-and-what-happens-next-134883 (Last accessed: 18.05.2020).

8. Usaid Siddiqui, Coronavirus testing methods: What you need to know, 07.04.2020 URL: https://www.aljazeera.com/news/2020/03/coronavirus-testing-methods-200330142718434.html (Last accessed: 18.05.2020).

9. Usaid Siddiqui, Timeline: How the new coronavirus spread 20.04.2020 https://www.aljazeera.com/news/2020/01/timeline-china-coronavirus-spread-200126061554884.html (Last accessed: 18.05.2020).

10. Jaziir O., Coronavirus Map 21.04.2020 URL: https://www.currenttime.tv/a/covid-19-interactive-map/30484955.html (Last accessed: 18.05.2020).

11. Юлия Хатеева, Актуальный список стран, в которых введен карантин по коронавирусу, 14.02.2020. URL: https://www.biletik.aero/handbook/blog/polnyy-spisok-stran-v-kotorykh-vveden-karantin-po-koronavirusu-obnovlyaetsya/ (Last accessed: 02.04.2020).

12. Jonathan C. L. Rodrigues, EditorialAn update on COVID-19 for the radiologist – A British society of Thoracic Imaging statement, 21.04.2020. URL:

https://www.researchgate.net/publication/340109534_An_update_on_COVID-19_for_the_radiologist_-_A_British_society_of_Thoracic_Imaging_statement

13. Brian Casey., How good is radiography for COVID-19 detection?, 21.04.2020. URL: https://www.auntminnie.com/index.aspx?sec=log&URL=https%3a%2f%2fwww.auntminnie.com%2findex.aspx%3fsec%3dsup%26sub%3dxra%26pag%3ddis%26ItemID%3d128618

14. Е.А.Пеликс., Принципы рентгенографии. Техника рентгенографического контроля, 11.04.2020. URL: https://www.spectroflash.ru/info/articles/287/x-ray_photography_principles/ (Last accessed: 17.04.2020).

15. Ілля Лячинский, Класифікація зображень за допомогою ЗНМ, 14.02.2019. URL: https://codeguida.com/post/1400 (Last acessed: 23.05.2020).

16. В. Т. Фисенко, Т. Ю. Фисенко., Компьютерная обработка и распознавание изображений: учеб. пособ. Санкт-Петербург СПбГУ ИТМО, 2008. 192 с.

17. Панченко Д.С., Путянин Е.П., Сравнительный анализ методов сегментации изображений 21.05.2020. URL: https://cyberleninka.ru/article/n/sravnitelnyy-analiz-metodov-segmentatsii-izobrazheniy/viewer (Last accessed 27.05.2020).

18. Geleone P., Dropout — метод решения проблемы переобучения в нейронных сетях, 28.04.2020. URL: https://habr.com/ru/company/wunderfund/blog/330814/ (Last accessed: 11.05.2020).

19. Хайкин С.., Нейронные сети, 22.04.2019. URL: https://towardsdatascience.com/review-squeezenet-image-classification-e7414825581a (Last accessed: 11.05.2020).

20. Sagar S., Activation Functions in Neural Networks, 16.09.2017. URL: https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6 (Last accessed: 22.05.2020).

21. Половинкин А. Н.., Алгоритмы классификации изображений с большим числом категорий объектов, 14.05.2013. URL: https://cyberleninka.ru/article/n/algoritmy-klassifikatsii-izobrazheniy-s-bolshim-chislom-kategoriy-obektov (Last accessed: 14.05.2020).

22. Недашківська Н. І., Конспект лекцій з дисципліни «Моделі і методи інтелектуального аналізу даних», 2019.

23. Nix S., Архитектуры нейросетей, 21.08.2018. URL: https://habr.com/ru/company/nix/blog/430524/ (дата звернення: 14.05.2020).

24. Dhillon A. & Verma K. G., “Progress in Artificial, Intelligence” , ‘ Convolutional neural network: a review of models, methodologies and applications to object detection’, vol.9, pp85–112, 2020.

25. Гудфеллоу Я., Бенджио И., Курвилл А., «Deep learning», vol.6-9, pp.150-314. 2017.

26. Литвинов С., ResNet (34, 50, 101): «остаточные» CNN для классификации изображений, 24.09.2019. URL: https://neurohive.io/ru/vidy-nejrosetej/resnet-34-50-101/ (Last accessed: 16.05.2020).

27. Forrest N. Iandola, Song Han, Matthew W. Moskewicz, Khalid Ashraf, William J. Dally, Kurt Keutzer, squeezenet: alexnet-level accuracy with 50x fewer parameters and <0.5mb model size", 2016.

28. Simonyan K., Zisserman A., Very deep convolutional networks for large-scale image recognition, 2016.

29. Жерон О., Прикладное машинное обучение с помощью Scikit-learn и TensorFlow, 2018. 239 с.

30. Sik-Ho Tsang., Review: SqueezeNet (Image Classification), 17.04.2017. URL:http://diggerdnepr.ddns.net/wpcontent/uploads/2019/05/khaykin_s_neyronnye_seti_polny_kurs_izd_2_2006_ru_pdf.pdf (Last accessed: 19.05.2020).

31. Калинин С.И., Сверточная сеть на python. Часть 3. Применение модели, 25.12.2017. URL: https://habr.com/ru/company/ods/blog/344888/ (Last accessed: 21.05.2020).

32. Жерон О., Прикладное машинное обучение с помощью Scikit-learn и TensorFlow, 2018. 239 с.

33. Хайкин С.., Нейронные сети, 22.04.2019. URL: https://towardsdatascience.com/review-squeezenet-image-classification-e7414825581a (Last accessed: 11.05.2020).

34. Brian Casey., Linear models, 21.04.2020. URL: https://scikit-learn.org/stable/modules/linear_model.html (Last accessed: 21.05.2020).

35. Слунский Л.Н., Анализ стабильности модели линейной регрессии во времени, 11.04.2018. URL: https://www.litres.ru/gettrial/?art=4958802&format=fb2&lfrom=204420127 (Last accessed: 17.05.2020).

36. Демиденко Е.З., Линейная и нелинейная регрессии, 14.02.2019. URL: https://scikit-learn.org/stable/modules/linear_model.html (Last acessed: 26.05.2020).

ДОДАТОК А ЛІСТИНГ ПРОГРАМ

Model_COVID19_CNN_VGG16.ipynb

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import VGG16
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.layers import AveragePooling2D
from tensorflow.keras.layers import Dropout
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Input
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.utils import to_categorical
from sklearn.preprocessing import LabelBinarizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from imutils import paths
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import random
import shutil
import cv2
import os


dataset_path = './dataset'


samples = 127

covid_dataset_path = 'covid-chest-xray'

csvPath = os.path.sep.join([covid_dataset_path, "metadata.csv"])
df = pd.read_csv(csvPath)

# loop over the rows of the COVID-19 data frame
for (i, row) in df.iterrows():
    # if (1) the current case is not COVID-19 or (2) this is not
    # a 'PA' view, then ignore the row
    if row["finding"] != "COVID-19" or row["view"] != "PA":
        continue

    # build the path to the input image file
    imagePath = os.path.sep.join([covid_dataset_path, "images", row["filename"]])

    # if the input image file does not exist (there are some errors in
    # the COVID-19 metadeta file), ignore the row
```

```
    if not os.path.exists(imagePath):
        continue

    # extract the filename from the image path and then construct the
    # path to the copied image file
    filename = row["filename"].split(os.path.sep)[-1]
    outputPath = os.path.sep.join([f"{dataset_path}/covid", filename])

    # copy the image
    shutil.copy2(imagePath, outputPath)

#%%

pneumonia_dataset_path ='chest_xray'

#%%

basePath = os.path.sep.join([pneumonia_dataset_path, "train", "NORMAL"])
imagePaths = list(paths.list_images(basePath))

# randomly sample the image paths
random.seed(42)
random.shuffle(imagePaths)
imagePaths = imagePaths[:samples]

# loop over the image paths
for (i, imagePath) in enumerate(imagePaths):
    # extract the filename from the image path and then construct the
    # path to the copied image file
    filename = imagePath.split(os.path.sep)[-1]
    outputPath = os.path.sep.join([f"{dataset_path}/normal", filename])

    # copy the image
    shutil.copy2(imagePath, outputPath)

#%%

def ceildiv(a, b):
    return -(-a // b)

def plots_from_files(imspaths, figsize=(10,5), rows=1, titles=None, maintitle=None):
    """Plot the images in a grid"""
    f = plt.figure(figsize=figsize)
    if maintitle is not None: plt.suptitle(maintitle, fontsize=10)
    for i in range(len(imspaths)):
        sp = f.add_subplot(rows, ceildiv(len(imspaths), rows), i+1)
        sp.axis('Off')
        if titles is not None: sp.set_title(titles[i], fontsize=16)
        img = plt.imread(imspaths[i])
        plt.imshow(img)

#%%
```

```
normal_images = list(paths.list_images(f"{dataset_path}/normal"))
covid_images = list(paths.list_images(f"{dataset_path}/covid"))

#%%

plots_from_files(normal_images, rows=5, maintitle="Normal X-ray images")

#%%

plots_from_files(covid_images, rows=5, maintitle="Covid-19 X-ray images")

#%%

# initialize the initial learning rate, number of epochs to train for,
# and batch size
INIT_LR = 1e-3
EPOCHS = 10
BS = 8

# grab the list of images in our dataset directory, then initialize
# the list of data (i.e., images) and class images
print("[INFO] loading images...")
imagePaths = list(paths.list_images(dataset_path))
data = []
labels = []
# loop over the image paths
for imagePath in imagePaths:
    # extract the class label from the filename
    label = imagePath.split(os.path.sep)[-2]
    # load the image, swap color channels, and resize it to be a fixed
    # 224x224 pixels while ignoring aspect ratio
    image = cv2.imread(imagePath)
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    image = cv2.resize(image, (224, 224))
    # update the data and labels lists, respectively
    data.append(image)
    labels.append(label)
# convert the data and labels to NumPy arrays while scaling the pixel
# intensities to the range [0, 1]
data = np.array(data) / 255.0
labels = np.array(labels)

#%%

# perform one-hot encoding on the labels
lb = LabelBinarizer()
labels = lb.fit_transform(labels)
labels = to_categorical(labels)
# partition the data into training and testing splits using 80% of
# the data for training and the remaining 20% for testing
```

```
(trainX, testX, trainY, testY) = train_test_split(data, labels, test_size=0.20, stratify=labels,
random_state=42)
# initialize the training data augmentation object
trainAug = ImageDataGenerator(rotation_range=15, fill_mode="nearest")

#%%

# load the VGG16 network, ensuring the head FC layer sets are left
# off
baseModel = VGG16(weights="imagenet", include_top=False, input_tensor=Input(shape=(224,
224, 3)))

headModel = baseModel.output
headModel = AveragePooling2D(pool_size=(4, 4))(headModel)
headModel = Flatten(name="flatten")(headModel)
headModel = Dense(64, activation="relu")(headModel)
headModel = Dropout(0.5)(headModel)
headModel = Dense(2, activation="softmax")(headModel)

model = Model(inputs=baseModel.input, outputs=headModel)

for layer in baseModel.layers:
    layer.trainable = False

#%%

# compile our model
print("[INFO] compiling model...")
opt = Adam(lr=INIT_LR, decay=INIT_LR / EPOCHS)
model.compile(loss="binary_crossentropy", optimizer=opt, metrics=["accuracy"])

# train the head of the network
print("[INFO] training head...")
H = model.fit_generator(
    trainAug.flow(trainX, trainY, batch_size=BS),
    steps_per_epoch=len(trainX) // BS,
    validation_data=(testX, testY),
    validation_steps=len(testX) // BS,
    epochs=EPOCHS)

#%%

# plot the training loss and accuracy
N = EPOCHS
plt.style.use("ggplot")
plt.figure()
plt.plot(np.arange(0, N), H.history["loss"], label="train_loss")
plt.plot(np.arange(0, N), H.history["val_loss"], label="val_loss")
plt.plot(np.arange(0, N), H.history["accuracy"], label="train_acc")
plt.plot(np.arange(0, N), H.history["val_accuracy"], label="val_acc")
plt.title("Training Loss and Accuracy on COVID-19 Dataset")
plt.xlabel("Epoch #")
```

```
plt.ylabel("Loss/Accuracy")
plt.legend(loc="lower left")
plt.savefig("plot.png")

#%%

# make predictions on the testing set
print("[INFO] evaluating network...")
predIdxs = model.predict(testX, batch_size=BS)
# for each image in the testing set we need to find the index of the
# label with corresponding largest predicted probability
predIdxs = np.argmax(predIdxs, axis=1)
# show a nicely formatted classification report
print(classification_report(testY.argmax(axis=1), predIdxs, target_names=lb.classes_))

#%%

# compute the confusion matrix and and use it to derive the raw
# accuracy, sensitivity, and specificity
cm = confusion_matrix(testY.argmax(axis=1), predIdxs)
total = sum(sum(cm))
acc = (cm[0, 0] + cm[1, 1]) / total
sensitivity = cm[0, 0] / (cm[0, 0] + cm[0, 1])
specificity = cm[1, 1] / (cm[1, 0] + cm[1, 1])
# show the confusion matrix, accuracy, sensitivity, and specificity
print(cm)
print("acc: {:.4f}".format(acc))
print("sensitivity: {:.4f}".format(sensitivity))
print("specificity: {:.4f}".format(specificity))

#%%

# Save the entire model to a HDF5 file.
# The '.h5' extension indicates that the model should be saved to HDF5.
model.save('covid_model.h5')

#%%

from tensorflow.keras.models import load_model
# Recreate the exact same model, including its weights and the optimizer
new_model = load_model('covid_model.h5')

# Show the model architecture
print('Model: VGG16_COVID19')
new_model.summary()


# make predictions on the testing set
print("[INFO] evaluating network...")
predIdxs = new_model.predict(testX, batch_size=BS)
# for each image in the testing set we need to find the index of the
# label with corresponding largest predicted probability
```

```
predIdxs = np.argmax(predIdxs, axis=1)
# show a nicely formatted classification report
print(classification_report(testY.argmax(axis=1), predIdxs, target_names=lb.classes_))


import numpy as np
from tensorflow.keras.preprocessing import image


#test_image = cv2.imread('../input/covid-chest-xray/images/1-s2.0-S1684118220300608-main.pdf-002.jpg')
#test_image = cv2.cvtColor(test_image, cv2.COLOR_BGR2RGB)
#test_image = cv2.resize(test_image, (224, 224))
#test_image = np.expand_dims(test_image, axis = 0)

img_width, img_height = 224, 224
img = image.load_img('covid-chest-xray/images/1.CXRCTThoraximagesofCOVID-19fromSingapore.pdf-003-fig4a.png', target_size = (img_width, img_height))
x = image.img_to_array(img)
img = np.expand_dims(x, axis = 0)

pred = new_model.predict(img)
print(pred)

#one = new_model.predict('../input/covid-chest-xray/images/1-s2.0-S1684118220300608-main.pdf-002.jpg', batch_size = BS)
# for each image in the testing set we need to find the index of the
# label with corresponding largest predicted probability
print(np.argmax(pred, axis=1))
if np.argmax(pred, axis=1)[0] == 1:
   plt.title('Prediction: Covid-19')
else:
   plt.title('Prediction: Non_Covid-19')
plt.imshow(x/255.)
plt.savefig('plot_out.png')


#%%

img_width, img_height = 224, 224
img = image.load_img('chest_xray/test/NORMAL/IM-0009-0001.jpeg', target_size = (img_width, img_height))
x = image.img_to_array(img)
img = np.expand_dims(x, axis = 0)

pred = new_model.predict(img)
print(pred)

#one = new_model.predict('../input/covid-chest-xray/images/1-s2.0-S1684118220300608-main.pdf-002.jpg', batch_size = BS)
# for each image in the testing set we need to find the index of the
# label with corresponding largest predicted probability
```

```
print(np.argmax(pred, axis=1)[0])
if np.argmax(pred, axis=1)[0] == 1:
   plt.title('Prediction: Non_Covid-19')
else:
   plt.title('Prediction: Covid-19')
plt.imshow(x/255.)

print(H.history["loss"])
loss = []

report = pd.DataFrame(columns = ['loss', 'acc', 'val_loss', 'val_acc'], index=['EPOCH 1','EPOCH
2','EPOCH 3','EPOCH 4','EPOCH 5','EPOCH 6','EPOCH 7','EPOCH 8','EPOCH 9','EPOCH 10'])
report['loss']  = H.history["loss"]
report['acc'] = H.history['accuracy']
report['val_loss'] = H.history["val_loss"]
report['val_acc'] = H.history['val_accuracy']

#%%

print(report)

Model_COVID19_CNN_RESNET50.ipynb

from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import VGG16
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.layers import AveragePooling2D
from tensorflow.keras.layers import Dropout
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Input
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.utils import to_categorical
from sklearn.preprocessing import LabelBinarizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from imutils import paths
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import random
import shutil
import cv2
import os

dataset_path = './dataset'

samples = 127

covid_dataset_path = 'covid-chest-xray'
```

```
#%%

# construct the path to the metadata CSV file and load it
csvPath = os.path.sep.join([covid_dataset_path, "metadata.csv"])
df = pd.read_csv(csvPath)

# loop over the rows of the COVID-19 data frame
for (i, row) in df.iterrows():
    # if (1) the current case is not COVID-19 or (2) this is not
    # a 'PA' view, then ignore the row
    if row["finding"] != "COVID-19" or row["view"] != "PA":
        continue

    # build the path to the input image file
    imagePath = os.path.sep.join([covid_dataset_path, "images", row["filename"]])

    # if the input image file does not exist (there are some errors in
    # the COVID-19 metadeta file), ignore the row
    if not os.path.exists(imagePath):
        continue

    # extract the filename from the image path and then construct the
    # path to the copied image file
    filename = row["filename"].split(os.path.sep)[-1]
    outputPath = os.path.sep.join([f"{dataset_path}/covid", filename])

    # copy the image
    shutil.copy2(imagePath, outputPath)

#%%

pneumonia_dataset_path ='chest_xray'

#%%

basePath = os.path.sep.join([pneumonia_dataset_path, "train", "NORMAL"])
imagePaths = list(paths.list_images(basePath))

# randomly sample the image paths
random.seed(42)
random.shuffle(imagePaths)
imagePaths = imagePaths[:samples]

# loop over the image paths
for (i, imagePath) in enumerate(imagePaths):
    # extract the filename from the image path and then construct the
    # path to the copied image file
    filename = imagePath.split(os.path.sep)[-1]
    outputPath = os.path.sep.join([f"{dataset_path}/normal", filename])

    # copy the image
```

```
    shutil.copy2(imagePath, outputPath)

#%%

def ceildiv(a, b):
    return -(-a // b)

def plots_from_files(imspaths, figsize=(10,5), rows=1, titles=None, maintitle=None):
    """Plot the images in a grid"""
    f = plt.figure(figsize=figsize)
    if maintitle is not None: plt.suptitle(maintitle, fontsize=10)
    for i in range(len(imspaths)):
        sp = f.add_subplot(rows, ceildiv(len(imspaths), rows), i+1)
        sp.axis('Off')
        if titles is not None: sp.set_title(titles[i], fontsize=16)
        img = plt.imread(imspaths[i])
        plt.imshow(img)


normal_images = list(paths.list_images(f"{dataset_path}/normal"))
covid_images = list(paths.list_images(f"{dataset_path}/covid"))


plots_from_files(normal_images, rows=5, maintitle="Normal X-ray images")


plots_from_files(covid_images, rows=5, maintitle="Covid-19 X-ray images")


# initialize the initial learning rate, number of epochs to train for,
# and batch size
INIT_LR = 1e-3
EPOCHS = 10
BS = 8

# grab the list of images in our dataset directory, then initialize
# the list of data (i.e., images) and class images
print("[INFO] loading images...")
imagePaths = list(paths.list_images(dataset_path))
data = []
labels = []
# loop over the image paths
for imagePath in imagePaths:
    # extract the class label from the filename
    label = imagePath.split(os.path.sep)[-2]
    # load the image, swap color channels, and resize it to be a fixed
    # 224x224 pixels while ignoring aspect ratio
    image = cv2.imread(imagePath)
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    image = cv2.resize(image, (224, 224))
    # update the data and labels lists, respectively
    data.append(image)
```

```
    labels.append(label)
# convert the data and labels to NumPy arrays while scaling the pixel
# intensities to the range [0, 1]
data = np.array(data) / 255.0
labels = np.array(labels)

#%%

# perform one-hot encoding on the labels
lb = LabelBinarizer()
labels = lb.fit_transform(labels)
labels = to_categorical(labels)
# partition the data into training and testing splits using 80% of
# the data for training and the remaining 20% for testing
(trainX, testX, trainY, testY) = train_test_split(data, labels, test_size=0.20, stratify=labels,
random_state=42)
# initialize the training data augmentation object
trainAug = ImageDataGenerator(rotation_range=15, fill_mode="nearest")

#%%



import ssl

ssl._create_default_https_context = ssl._create_unverified_contextbaseModel =
ResNet50(weights="imagenet", include_top=False, input_tensor=Input(shape=(224, 224, 3)))
# construct the head of the model that will be placed on top of the
# the base model
headModel = baseModel.output
headModel = AveragePooling2D(pool_size=(4, 4))(headModel)
headModel = Flatten(name="flatten")(headModel)
headModel = Dense(64, activation="relu")(headModel)
headModel = Dropout(0.5)(headModel)
headModel = Dense(2, activation="softmax")(headModel)
# place the head FC model on top of the base model (this will become
# the actual model we will train)
model = Model(inputs=baseModel.input, outputs=headModel)
# loop over all layers in the base model and freeze them so they will
# *not* be updated during the first training process
for layer in baseModel.layers:
    layer.trainable = False

#%%

model_resnet = model

#%%

# compile our model
print("[INFO] compiling model...")
```

```
opt = Adam(lr=INIT_LR, decay=INIT_LR / EPOCHS)
model_resnet.compile(loss="binary_crossentropy", optimizer=opt, metrics=["accuracy"])

# train the head of the network
print("[INFO] training head...")
H = model_resnet.fit(
    trainAug.flow(trainX, trainY, batch_size=BS),
    steps_per_epoch=len(trainX) // BS,
    validation_data=(testX, testY),
    validation_steps=len(testX) // BS,
    epochs=EPOCHS)

#%%

# plot the training loss and accuracy
N = EPOCHS
plt.style.use("ggplot")
plt.figure()
plt.plot(np.arange(0, N), H.history["loss"], label="train_loss")
plt.plot(np.arange(0, N), H.history["val_loss"], label="val_loss")
plt.plot(np.arange(0, N), H.history["accuracy"], label="train_acc")
plt.plot(np.arange(0, N), H.history["val_accuracy"], label="val_acc")
plt.title("Training Loss and Accuracy on COVID-19 Dataset")
plt.xlabel("Epoch #")
plt.ylabel("Loss/Accuracy")
plt.legend(loc="lower left")
plt.savefig("plot.png")

#%%

# make predictions on the testing set
print("[INFO] evaluating network...")
predIdxs = model.predict(testX, batch_size=BS)
# for each image in the testing set we need to find the index of the
# label with corresponding largest predicted probability
predIdxs = np.argmax(predIdxs, axis=1)
# show a nicely formatted classification report
print(classification_report(testY.argmax(axis=1), predIdxs, target_names=lb.classes_))

#%%

# compute the confusion matrix and and use it to derive the raw
# accuracy, sensitivity, and specificity
cm = confusion_matrix(testY.argmax(axis=1), predIdxs)
total = sum(sum(cm))
acc = (cm[0, 0] + cm[1, 1]) / total
sensitivity = cm[0, 0] / (cm[0, 0] + cm[0, 1])
specificity = cm[1, 1] / (cm[1, 0] + cm[1, 1])
# show the confusion matrix, accuracy, sensitivity, and specificity
print(cm)
print("acc: {:.4f}".format(acc))
print("sensitivity: {:.4f}".format(sensitivity))
```

```
print("specificity: {:.4f}".format(specificity))

#%%

# Save the entire model to a HDF5 file.
# The '.h5' extension indicates that the model should be saved to HDF5.
model_resnet.save('covid_resnet_model.h5')

#%%

from tensorflow.keras.models import load_model
# Recreate the exact same model, including its weights and the optimizer
new_model = load_model('covid_resnet_model.h5')

# Show the model architecture
new_model.summary()

#%%

# make predictions on the testing set
print("[INFO] evaluating network...")
predIdxs = new_model.predict(testX, batch_size=BS)
# for each image in the testing set we need to find the index of the
# label with corresponding largest predicted probability
predIdxs = np.argmax(predIdxs, axis=1)
# show a nicely formatted classification report
print(classification_report(testY.argmax(axis=1), predIdxs, target_names=lb.classes_))

#%%

import numpy as np
from tensorflow.keras.preprocessing import image


#test_image = cv2.imread('../input/covid-chest-xray/images/1-s2.0-S1684118220300608-
main.pdf-002.jpg')
#test_image = cv2.cvtColor(test_image, cv2.COLOR_BGR2RGB)
#test_image = cv2.resize(test_image, (224, 224))
#test_image = np.expand_dims(test_image, axis = 0)

img_width, img_height = 224, 224
img = image.load_img('covid-chest-xray/images/1-s2.0-S0929664620300449-gr2_lrg-b.jpg',
target_size = (img_width, img_height))
x = image.img_to_array(img)
img = np.expand_dims(x, axis = 0)

pred = new_model.predict(img)
print(pred)

#one = new_model.predict('../input/covid-chest-xray/images/1-s2.0-S1684118220300608-
main.pdf-002.jpg', batch_size = BS)
# for each image in the testing set we need to find the index of the
```

```
# label with corresponding largest predicted probability
print(np.argmax(pred, axis=1))
if np.argmax(pred, axis=1)[0] == 1:
   plt.title('Prediction: Covid-19')
else:
   plt.title('Prediction: Non_Covid-19')
plt.imshow(x/255.)
plt.savefig('plot_out.png')


#%%

img_width, img_height = 224, 224
img = image.load_img('chest_xray/test/NORMAL/IM-0017-0001.jpeg', target_size =
(img_width, img_height))
x = image.img_to_array(img)
img = np.expand_dims(x, axis = 0)

pred = new_model.predict(img)
print(pred)

#one = new_model.predict('../input/covid-chest-xray/images/1-s2.0-S1684118220300608-
main.pdf-002.jpg', batch_size = BS)
# for each image in the testing set we need to find the index of the
# label with corresponding largest predicted probability
print(np.argmax(pred, axis=1)[0])
if np.argmax(pred, axis=1)[0] == 1:
   plt.title('Prediction: Non_Covid-19')
else:
   plt.title('Prediction: Covid-19')
plt.imshow(x/255.)

#%%

report = pd.DataFrame(columns = ['loss', 'acc', 'val_loss', 'val_acc'], index=['EPOCH 1','EPOCH
2','EPOCH 3','EPOCH 4','EPOCH 5','EPOCH 6','EPOCH 7','EPOCH 8','EPOCH 9','EPOCH 10',
'EPOCH 11','EPOCH 12','EPOCH 13','EPOCH 14','EPOCH 15'])
report['loss']  = H.history["loss"]
report['acc'] = H.history['accuracy']
report['val_loss'] = H.history["val_loss"]
report['val_acc'] = H.history['val_accuracy']

#%%

print(report)

Squeezenet_model_covid19.ipynb

#%%

from sklearn.preprocessing import LabelBinarizer
from sklearn.model_selection import train_test_split
```

```
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from imutils import paths
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import random
import shutil
import cv2
import os
import tensorflow as tf

#%%

dataset_path = './sq_dataset'

#%%

samples = 125

#%%

covid_dataset_path = 'covid-chest-xray'

#%%

# construct the path to the metadata CSV file and load it
csvPath = os.path.sep.join([covid_dataset_path, "metadata.csv"])
df = pd.read_csv(csvPath)

# loop over the rows of the COVID-19 data frame
for (i, row) in df.iterrows():
    # if (1) the current case is not COVID-19 or (2) this is not
    # a 'PA' view, then ignore the row
    if row["finding"] != "COVID-19" or row["view"] != "PA":
        continue

    # build the path to the input image file
    imagePath = os.path.sep.join([covid_dataset_path, "images", row["filename"]])

    # if the input image file does not exist (there are some errors in
    # the COVID-19 metadeta file), ignore the row
    if not os.path.exists(imagePath):
        continue

    # extract the filename from the image path and then construct the
    # path to the copied image file
    filename = row["filename"].split(os.path.sep)[-1]
    outputPath = os.path.sep.join([f"{dataset_path}/covid", filename])

    # copy the image
    shutil.copy2(imagePath, outputPath)
```

```
#%%

pneumonia_dataset_path ='chest_xray'
basePath = os.path.sep.join([pneumonia_dataset_path, "train", "PNEUMONIA"])
imagePaths = list(paths.list_images(basePath))

# randomly sample the image paths
random.seed(42)
random.shuffle(imagePaths)
imagePaths = imagePaths[:samples]

# loop over the image paths
for (i, imagePath) in enumerate(imagePaths):
    # extract the filename from the image path and then construct the
    # path to the copied image file
    filename = imagePath.split(os.path.sep)[-1]
    outputPath = os.path.sep.join([f"{dataset_path}/covid", filename])

    # copy the image
    shutil.copy2(imagePath, outputPath)

#%%

pneumonia_dataset_path ='chest_xray'
basePath = os.path.sep.join([pneumonia_dataset_path, "train", "NORMAL"])
imagePaths = list(paths.list_images(basePath))

# randomly sample the image paths
random.seed(42)
random.shuffle(imagePaths)
imagePaths = imagePaths[:samples]

# loop over the image paths
for (i, imagePath) in enumerate(imagePaths):
    # extract the filename from the image path and then construct the
    # path to the copied image file
    filename = imagePath.split(os.path.sep)[-1]
    outputPath = os.path.sep.join([f"{dataset_path}/normal", filename])

    # copy the image
    shutil.copy2(imagePath, outputPath)

#%%

def ceildiv(a, b):
    return -(-a // b)

def plots_from_files(imspaths, figsize=(10,5), rows=1, titles=None, maintitle=None):
    """Plot the images in a grid"""
    f = plt.figure(figsize=figsize)
    if maintitle is not None: plt.suptitle(maintitle, fontsize=10)
```

```
    for i in range(len(imspaths)):
        sp = f.add_subplot(rows, ceildiv(len(imspaths), rows), i+1)
        sp.axis('Off')
        if titles is not None: sp.set_title(titles[i], fontsize=16)
        img = plt.imread(imspaths[i])
        plt.imshow(img)

#%%

normal_images = list(paths.list_images(f"{dataset_path}/normal"))
covid_images = list(paths.list_images(f"{dataset_path}/covid"))

#%%

plots_from_files(normal_images, rows=5, maintitle="Normal X-ray images")

#%%

plots_from_files(covid_images, rows=5, maintitle="Covid-19 X-ray images")

#%%

# initialize the initial learning rate, number of epochs to train for,
# and batch size
INIT_LR = 1e-3
EPOCHS = 10
BS = 8


#%%

# grab the list of images in our dataset directory, then initialize
# the list of data (i.e., images) and class images
print("[INFO] loading images...")
imagePaths = list(paths.list_images(dataset_path))
data = []
labels = []
# loop over the image paths
for imagePath in imagePaths:
    # extract the class label from the filename
    label = imagePath.split(os.path.sep)[-2]
    # load the image, swap color channels, and resize it to be a fixed
    # 224x224 pixels while ignoring aspect ratio
    image = cv2.imread(imagePath)
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    image = cv2.resize(image, (224, 224))
    # update the data and labels lists, respectively
    data.append(image)
    labels.append(label)
# convert the data and labels to NumPy arrays while scaling the pixel
# intensities to the range [0, 1]
data = np.array(data) / 255.0
```

```
labels = np.array(labels)
# print(labels)

#%%

from tensorflow.keras.utils import to_categorical
from tensorflow.keras.preprocessing.image import ImageDataGenerator
# perform one-hot encoding on the labels
lb = LabelBinarizer()
labels = lb.fit_transform(labels)
labels = to_categorical(labels)
# partition the data into training and testing splits using 80% of
# the data for training and the remaining 20% for testing
(trainX, testX, trainY, testY) = train_test_split(data, labels, test_size=0.20, stratify=labels,
random_state=42)
# initialize the training data augmentation object
trainAug = ImageDataGenerator(rotation_range=15, fill_mode="nearest")

#%%

# Detect hardware
try:
  tpu = tf.distribute.cluster_resolver.TPUClusterResolver() # TPU detection
except ValueError:
  tpu = None
  gpus = tf.config.experimental.list_logical_devices("GPU")

# Select appropriate distribution strategy for hardware
if tpu:
  tf.config.experimental_connect_to_cluster(tpu)
  tf.tpu.experimental.initialize_tpu_system(tpu)
  strategy = tf.distribute.experimental.TPUStrategy(tpu)
  print('Running on TPU ', tpu.master())
elif len(gpus) > 0:
  strategy = tf.distribute.MirroredStrategy(gpus) # this works for 1 to multiple GPUs
  print('Running on ', len(gpus), ' GPU(s) ')
else:
  strategy = tf.distribute.get_strategy() # default strategy that works on CPU and single GPU
  print('Running on CPU')

# How many accelerators do we have ?
print("Number of accelerators: ", strategy.num_replicas_in_sync)



#%%

with strategy.scope(): # this line is all that is needed to run on TPU (or multi-GPU, ...)

  bnmomemtum=0.9
  def fire(x, squeeze, expand):
```

```
    y  = tf.keras.layers.Conv2D(filters=squeeze, kernel_size=1, activation='relu',
padding='same')(x)
    y = tf.keras.layers.BatchNormalization(momentum=bnmomemtum)(y)
    y1 = tf.keras.layers.Conv2D(filters=expand//2, kernel_size=1, activation='relu',
padding='same')(y)
    y1 = tf.keras.layers.BatchNormalization(momentum=bnmomemtum)(y1)
    y3 = tf.keras.layers.Conv2D(filters=expand//2, kernel_size=3, activation='relu',
padding='same')(y)
    y3 = tf.keras.layers.BatchNormalization(momentum=bnmomemtum)(y3)
    return tf.keras.layers.concatenate([y1, y3])

  def fire_module(squeeze, expand):
    return lambda x: fire(x, squeeze, expand)

  x = tf.keras.layers.Input(shape=[224, 224, 3]) # input is 192x192 pixels RGB

  y = tf.keras.layers.Conv2D(kernel_size=3, filters=32, padding='same', use_bias=True,
activation='relu')(x)
  y = tf.keras.layers.BatchNormalization(momentum=bnmomemtum)(y)
  y = fire_module(24, 48)(y)
  y = tf.keras.layers.MaxPooling2D(pool_size=2)(y)
  y = fire_module(48, 96)(y)
  y = tf.keras.layers.MaxPooling2D(pool_size=2)(y)
  y = fire_module(64, 128)(y)
  y = tf.keras.layers.MaxPooling2D(pool_size=2)(y)
  y = fire_module(48, 96)(y)
  y = tf.keras.layers.MaxPooling2D(pool_size=2)(y)
  y = fire_module(24, 48)(y)
  y = tf.keras.layers.GlobalAveragePooling2D()(y)
  y = tf.keras.layers.Dropout(0.5)(y)
  y = tf.keras.layers.Dense(2, activation='softmax')(y)

  model = tf.keras.Model(x, y)

  model.compile(
    optimizer='adam',
    loss= 'binary_crossentropy',
    metrics=['accuracy'])

  model.summary()



#%%


H = model.fit(
    trainAug.flow(trainX, trainY, batch_size=BS),
    steps_per_epoch=len(trainX) // BS,
    validation_data=(testX, testY),
    validation_steps=len(testX) // BS,
```

```
    epochs=EPOCHS)

#%%

# plot the training loss and accuracy
N = EPOCHS
plt.style.use("ggplot")
plt.figure()
plt.plot(np.arange(0, N), H.history["loss"], label="train_loss")
plt.plot(np.arange(0, N), H.history["val_loss"], label="val_loss")
plt.plot(np.arange(0, N), H.history["accuracy"], label="train_acc")
plt.plot(np.arange(0, N), H.history["val_accuracy"], label="val_acc")
plt.title("Training Loss and Accuracy on COVID-19 Dataset")
plt.xlabel("Epoch #")
plt.ylabel("Loss/Accuracy")
plt.legend(loc="lower left")
plt.savefig("plot.png")

#%%

# make predictions on the testing set
print("[INFO] evaluating network...")
predIdxs = model.predict(testX, batch_size=BS)
# for each image in the testing set we need to find the index of the
# label with corresponding largest predicted probability
predIdxs = np.argmax(predIdxs, axis=1)
# show a nicely formatted classification report
print(classification_report(testY.argmax(axis=1), predIdxs, target_names=lb.classes_))

#%%

# compute the confusion matrix and and use it to derive the raw
# accuracy, sensitivity, and specificity
cm = confusion_matrix(testY.argmax(axis=1), predIdxs)
total = sum(sum(cm))
acc = (cm[0, 0] + cm[1, 1]) / total
sensitivity = cm[0, 0] / (cm[0, 0] + cm[0, 1])
specificity = cm[1, 1] / (cm[1, 0] + cm[1, 1])
# show the confusion matrix, accuracy, sensitivity, and specificity
print(cm)
print("acc: {:.4f}".format(acc))
print("sensitivity: {:.4f}".format(sensitivity))
print("specificity: {:.4f}".format(specificity))

#%%

model.save('covid_model_sqz.h5')

#%%

from tensorflow.keras.models import load_model
# Recreate the exact same model, including its weights and the optimizer
```

```
new_model = load_model('covid_model_sqz.h5')

# Show the model architecture
new_model.summary()

#%%

# make predictions on the testing set
print("[INFO] evaluating network...")
predIdxs = new_model.predict(testX, batch_size=BS)
# for each image in the testing set we need to find the index of the
# label with corresponding largest predicted probability
predIdxs = np.argmax(predIdxs, axis=1)
# show a nicely formatted classification report
print(classification_report(testY.argmax(axis=1), predIdxs, target_names=lb.classes_))

#%%

import numpy as np
from tensorflow.keras.preprocessing import image


#test_image = cv2.imread('../input/covid-chest-xray/images/1-s2.0-S1684118220300608-main.pdf-002.jpg')
#test_image = cv2.cvtColor(test_image, cv2.COLOR_BGR2RGB)
#test_image = cv2.resize(test_image, (224, 224))
#test_image = np.expand_dims(test_image, axis = 0)

img_width, img_height = 224, 224
# img = image.load_img('covid-chest-xray/images/8FDE8DBA-CFBD-4B4C-B1A4-6F36A93B7E87.jpeg', target_size = (img_width, img_height))
img = image.load_img('chest_xray/test/PNEUMONIA/person66_virus_125.jpeg', target_size = (img_width, img_height))
x = image.img_to_array(img)
img = np.expand_dims(x, axis = 0)

pred = new_model.predict(img)
print(pred)

#one = new_model.predict('../input/covid-chest-xray/images/1-s2.0-S1684118220300608-main.pdf-002.jpg', batch_size = BS)
# for each image in the testing set we need to find the index of the
# label with corresponding largest predicted probability
print(np.argmax(pred, axis=1))
if np.argmax(pred, axis=1)[0] == 1:
   plt.title('Prediction: Covid-19')
else:
   plt.title('Prediction: Non_Covid-19')
plt.imshow(x/255.)
plt.savefig('plot_out.png')

#%%
```

```
img_width, img_height = 224, 224
img = image.load_img('chest_xray/test/NORMAL/IM-0015-0001.jpeg', target_size =
(img_width, img_height))
x = image.img_to_array(img)
img = np.expand_dims(x, axis = 0)

pred = new_model.predict(img)
print(pred)

#one = new_model.predict('../input/covid-chest-xray/images/1-s2.0-S1684118220300608-
main.pdf-002.jpg', batch_size = BS)
# for each image in the testing set we need to find the index of the
# label with corresponding largest predicted probability
print(np.argmax(pred, axis=1)[0])
if np.argmax(pred, axis=1)[0] == 1:
   plt.title('Prediction: Non_Covid-19')
else:
   plt.title('Prediction: Covid-19')
plt.imshow(x/255.)
```

MLRegression_Model+SVM_model.ipynb

```
# https://scikit-
learn.org/stable/modules/generated/sklearn.neural_network.MLPRegressor.html?highlight=mlp#
sklearn.neural_network.MLPRegressor
# https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html
# https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
from datetime import date, timedelta
from sklearn.metrics import accuracy_score, classification_report
path = 'coronavirusdataset/train.csv'
df = pd.read_csv(path)
df.head()
# type casting : object -> datetime
df.confirmed_date = pd.to_datetime(df.confirmed_date)

# get daily confirmed count
# daily_count = df.groupby(df.confirmed_date).ConfirmedCases.count()
world_df = pd.DataFrame(df, columns=['confirmed_date', 'ConfirmedCases'])
world_df = world_df.groupby(df.confirmed_date).agg({'ConfirmedCases':'sum'})

daily = []
for i in range(0,115):
   if i == 0:
      day = world_df['ConfirmedCases'][i]
   day = world_df['ConfirmedCases'][i] - world_df['ConfirmedCases'][i-1]
   daily.append(day)
```

```
daily[0] = 554
world_df_daily = world_df
world_df_daily['daily_count'] = daily
world_df_daily.head()


# get accumulated confirmed count
# accumulated_count = daily_count.cumsum()
# функция для создания дата фрейма по заданной стране с комулитативной суммой, и с дневным количеством, передаем в нее название страны
# в виде строки и основную df
def df_country(country,df):
    country_df = df[(df['Country_Region']==country)]
    country_df = pd.DataFrame(country_df, columns=['confirmed_date', 'ConfirmedCases'])
    country_df = country_df.groupby(df.confirmed_date).agg({'ConfirmedCases':'sum'})
    daily = []
    for i in range(0,len(list(country_df['ConfirmedCases']))):
        day = country_df['ConfirmedCases'][i] - country_df['ConfirmedCases'][i-1]
        daily.append(day)
    country_df['daily_count'] = daily
    country_df['daily_count'][0] = country_df['ConfirmedCases'][0]
    return country_df



def plot_daily_and_sum(df):
    plt.figure(figsize=(17,10))
    plt.subplot(2, 2, 1)
    df['daily_count'].plot(ax=plt.gca(), title='Daily Confirmed Count')
    plt.ylabel("Confirmed infection cases", size=13)

    plt.subplot(2, 2, 2)
    df['ConfirmedCases'].plot(ax=plt.gca(), title='Comulative Sum')

plot_daily_and_sum(world_df_daily)
Italy = df_country('Italy', df)
plot_daily_and_sum(Italy)
Ukraine = df_country('Ukraine', df)
plot_daily_and_sum(Ukraine)
Russia = df_country('Russia', df)
plot_daily_and_sum(Russia)
Australia = df_country('Australia', df)
plot_daily_and_sum(Australia)
def train_and_test(df):
    pick_date = df[(df['daily_count'] == df['daily_count'].max())].index[0]
    test_start_date = pick_date -  timedelta(days=7)
    train_start_date = test_start_date -  timedelta(days=30)
    train = [train_start_date,(test_start_date-timedelta(days=1))]
    test = [test_start_date,pick_date]
    return train, test
from sklearn.neural_network import MLPRegressor
from sklearn.metrics import r2_score
```

```
def MLPR_predict(df):
    idx = train_and_test(df)
    train = idx[0]
    test = idx[1]
    test_1 = test[0]
    test_2 = test[1] - timedelta(days=1)
    train_data = df[train[0]:train[1]]
    x = np.arange(len(train_data)).reshape(-1, 1)
    y = train_data['ConfirmedCases'].values
#     MLPRegressor(hidden_layer_sizes=(100, ), activation='relu', *, solver='adam',
alpha=0.0001, batch_size='auto', learning_rate='constant', learning_rate_init=0.001,
power_t=0.5, max_iter=200, shuffle=True, random_state=None, tol=0.0001, verbose=False,
warm_start=False, momentum=0.9, nesterovs_momentum=True, early_stopping=False,
validation_fraction=0.1, beta_1=0.9, beta_2=0.999, epsilon=1e-08, n_iter_no_change=10,
max_fun=15000)
    model = MLPRegressor(hidden_layer_sizes=[32, 26, 10, ], max_iter=100000, alpha=0.0005,
random_state=26,  solver='lbfgs',  learning_rate='constant', validation_fraction=0.1)
    _=model.fit(x, y)
    test = np.arange(len(train_data)+7).reshape(-1, 1)
    pred = model.predict(test)
    prediction = pred.round().astype(int)
    week = [train_data.index[0] + timedelta(days=i) for i in range(len(prediction))]
    dt_idx = pd.DatetimeIndex(week)
    predicted_count = pd.Series(prediction, dt_idx)
    train_data['ConfirmedCases'].plot()
    predicted_count[-8:].plot()
    print('Predicted data: ')
    print(predicted_count[-7:])
    print('Real data: ')
    print(df[test_1:test_2]['ConfirmedCases'])
#     print(df[test_1:test_2])
#     print('accuracy: ', str((accuracy_score(train_data['ConfirmedCases'], predicted_count[0:30],
normalize=False))*100), ' %')
    plt.title('Prediction of Accumulated Confirmed Count')
    plt.legend(['current confirmd count', 'predicted confirmed count'])
    plt.show()
    print(r2_score(predicted_count[-7:].values, df[test_1:test_2]['ConfirmedCases'].values))

MLPR_predict(df_country('Ukraine',df))

from sklearn.svm import SVR
from sklearn.metrics import r2_score
def SVR_predict(df):
    idx = train_and_test(df)
    train = idx[0]
    test = idx[1]
    test_1 = test[0]
    test_2 = test[1] - timedelta(days=1)
    train_data = df[train[0]:train[1]]
    x = np.arange(len(train_data)).reshape(-1,1)
    y = train_data['ConfirmedCases'].values
```

```
#     SVR(*, kernel='rbf', degree=3, gamma='scale', coef0=0.0, tol=0.001, C=1.0, epsilon=0.1,
shrinking=True, cache_size=200, verbose=False, max_iter=-1)
    model = SVR(kernel='poly', degree=3, C=30, epsilon=0.1, shrinking=True, cache_size=200,
verbose=False, tol=0.001, max_iter=-1)
    _=model.fit(x, y)
    test = np.arange(len(train_data)+7).reshape(-1, 1)
    pred = model.predict(test)
    prediction = pred.round().astype(int)
    week = [train_data.index[0] + timedelta(days=i) for i in range(len(prediction))]
    dt_idx = pd.DatetimeIndex(week)
    predicted_count = pd.Series(prediction, dt_idx)
    train_data['ConfirmedCases'].plot()
    predicted_count.plot()
    print(predicted_count[-7:])
    print(df[test_1:test_2])
#     print('accuracy: ', str((accuracy_score(train_data['ConfirmedCases'], predicted_count[0:30],
normalize=False))*100), ' %')
    plt.title('Prediction of Accumulated Confirmed Count')
    plt.legend(['current confirmd count', 'predicted confirmed count'])
    plt.show()
    print(r2_score(predicted_count[-7:].values, df[test_1:test_2]['ConfirmedCases'].values))

SVR_predict(df_country('Mexico',df))



from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor,
AdaBoostRegressor, BaggingRegressor, VotingRegressor, StackingRegressor

from sklearn.model_selection import train_test_split



def trainModels(df):
    # split into train and test
    idx = train_and_test(df)
    train = idx[0]
    test = idx[1]
    test_1 = test[0]
    test_2 = test[1]
    train_data = df[train[0]:train[1]]
    test_data = df[test_1:test_2]
    x = np.arange(len(train_data)).reshape(-1,1)
    y = train_data['daily_count'].values
    test = np.arange(len(test_data)).reshape(-1, 1)
    y2 = test_data['daily_count'].values
    pick_date = df[(df['daily_count'] == df['daily_count'].max())].index[0]
#     start_date -  timedelta(days=30)
    # split into train and test
#     train, test = train_test_split(df['daily_count'][:pick_date], test_size=0.2)
```

```
    # Removing the target/predictor from the train data
#     targ = train[list(target_col)]

#     imp_attr = ['confirmed_date']
#     target_col = ["daily_count"]
    ### Random Forest Model
    rand_forest_model = RandomForestRegressor(n_estimators=1000, max_features=1,
oob_score=True, random_state=115)
    rand_forest_model.fit(x,y)
    print('R-square value of Random Forest Model: ', rand_forest_model.score(test, y2))

    ### Gradient Boosting
    gr_boosting_model = GradientBoostingRegressor(n_estimators=500, learning_rate=0.1,
subsample=1, max_features=1, loss='ls')
    gr_boosting_model.fit(x,y)
    print('Gradient Boosting:', gr_boosting_model.score(test, y2))

    # Ada Boosting Regressor
    ad_boost_model = AdaBoostRegressor(base_estimator=None, n_estimators=50,
learning_rate=1.0, loss='linear',
                                       random_state=None)
    ad_boost_model.fit(x,y)
    print('Ada Boosting Regressor:', ad_boost_model.score(test, y2))

    # Bagging Regressor
    bagging_model = BaggingRegressor(base_estimator=None, n_estimators=10,
max_samples=1.0, max_features=1.0,
                                     bootstrap=True, bootstrap_features=False, oob_score=False,
warm_start=False, n_jobs=None, random_state=None, verbose=0)
    bagging_model.fit(x,y)
    print('Bagging Regressor', bagging_model.score(test, y2))

    # Voting Regressor
    voting_model = VotingRegressor(estimators=100, weights=None, n_jobs=None,
verbose=False)
    voting_model.fit(x,y)
    print('Voting Regressor', voting_model.score(test, y2))

    # Stacking Regressor
    stacking_model = StackingRegressor(estimators=100, final_estimator=None, cv=None,
n_jobs=None,
                                       passthrough=False, verbose=0)
    stacking_model.fit(x,y)
    print('Stacking Regressor', stacking_model.score(test, y2))

trainModels(df_country('Belgium',df))
```

ДОДАТОК Б ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

Моделі та методи інтелектуального аналізу даних COVID-19. Прогнозування діагнозу та виявлення факторів, які впливають на перебіг захворювання

Виконав: Сапєльніков Олександр Сергійович

Науковий керівний:
д.т.н. доцент Недашківська Надія Іванівна

Постановка задачі

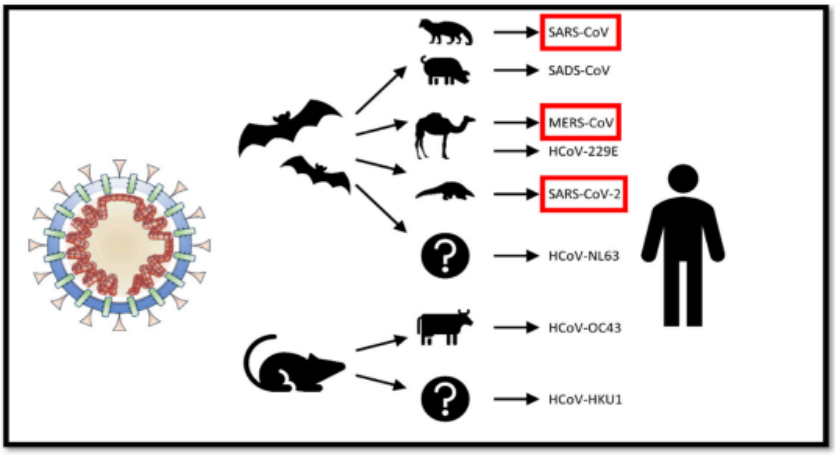
- Провести дослідження та проаналізувати ситуацію з поширенням коронавірусу в різних країнах світу
- Провести аналіз методів класифікації зображень
- Побудувати моделі класифікації зображень рентгенів здорової людини і хворої на COVID19
- Виконати прогнозування діагнозу COVID19 на основі вибраної найкращої моделі
- Виявити фактори, які впливають на перебіг захворювання
- Розробити програмний продукт для прогнозування діагнозу COVID19 на основі вибраної найкращої моделі

Актуальність роботи

COVID-19 досить має великий потенціал розвитку та поширення, не виключається варіант другої хвилі захворювання а також що ця інфекція є сезонною, і якщо її не подолають повністю, то вже в осені ми можемо зустріти нову хвилю. Саме через це, думки багатьох вчених зараз направлені на те, щоб знаходити нові та кращі варіанти виявлення хвороби у людини якомога швидше, враховуючи не надійність існуючих тестів.

Один з таких варіантів виявлення, це розпізнавання ознак хвороби на рентген знімках грудної клітини. Недавні дослідження показали, що рентген, поступаючись КТ в точності діагностики, проте, може зіграти свою роль в приборканні пандемії коронавіруса.

Походження коронавірусу



Що таке коронавіруси

Коронавіруси – це віруси, які циркулюють серед тварин, а деякі з них також, як відомо, заражають людину. На сьогодні відомо також те, що кілька видів тварин виступають джерелами. У людей, зазвичай, вони викликають респіраторні інфекції, починаючи від звичайної застуди до більш важких захворювань. Останній виявлений коронавірус викликає – COVID-19. Абревіатуру обрала ВООЗ, котра означає «коронавірусна хвороба 2019».

Рентген зображення

Це зображення отриманне в наслідку дослідження внутрішньої структури об'єктів, які проектуються за допомогою рентгеновскіх променів на спеціальну плівку або папір. Найбільш часто термін відноситься до медичного неінвазивного дослідження, заснованого на отриманні сумарного проекційного зображення анатомічних структур організму за допомогою проходження через них рентгенівських променів і реєстрації ступеня ослаблення рентгенівського випромінювання.

Приклад рентген зображень

Здорової людини

Хворої на COVID-19

Порівняльний аналіз методів класифікації зображень

Метод сегментації на основі аналізу різниці яскравості

- Пороговий метод
- Виділення границь
- Аналіз кольору
- Метод нарощення областей

Текстурні методи сегментації

- Матриця співпадінь
- Метод мозаїки Вороного

Згорткові нейронні мережі (CNN)

Convolution Neural Network (CNN) з'явилися в результаті вивчення зорової кори головного мозку і застосовувалися в розпізнаванні зображень, починаючи з 1980-х років.

Найважливіший будівельний блок мережі CNN – це згортковий шар: нейрони в першому згортковому шарі не пов'язані з кожним поодиноким пікселем у вхідному зображенні , а тільки з пікселями в власних рецепторних полях. У свою чергу кожен нейрон у другому згортковому шарі пов'язаний тільки з нейронами, що знаходяться всередині невеликого прямокутника в першому шарі.

Згортка

$$s(t) = \int u(a)w(t-a)da,$$

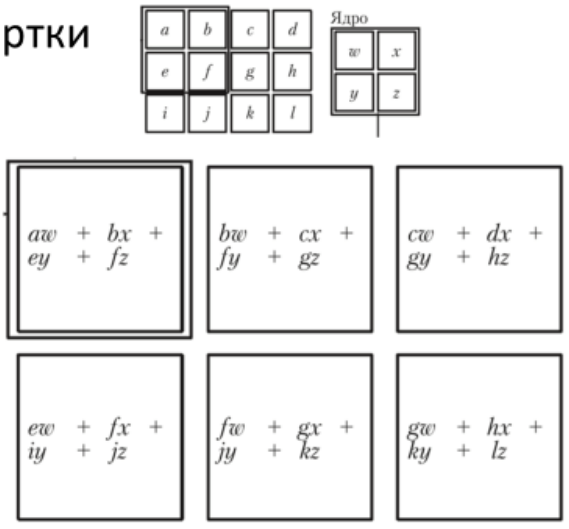
$$s(t) = (u * w)(t).$$

Згортка - це особливий вид лінійної операції над двома функціями дійсного аргументу.
Приклад: датчик, який видає єдине значення u(t) - це положення корабля в момент t.

виміри датчика містять шум,

1. для отримання більш точної оцінки положення корабля візьмемо декілька результатів вимірів і усереднимо їх,
2. останні виміри більш важливі, тому розраховуємо зважене середнє, надаючи більшу вагу останнім вимірам,
3. візьмемо вагову функцію w(a), де a - давність виміру
4. Отримаємо нову функцію, яка дає згладжену оцінку положення корабля і називається згорткою.

Приклад згортки



Скриті шари ReLU

CNN складається з:

- вхідного шару,
- вихідного шару,
- множини прихованих шарів: згорткові шари, об'єднання шарів, повно зв’язних шарів і шарів нормалізації (ReLU).

Нормалізаційний шар CNN, також називають процесом випрямленого лінійного блоку (ReLU):

$$f(x) = x^+ = \max(0, x)$$

x - вхід до нейрона.

Вихідний шар SoftMax

Функція SoftMax:

$$softmax(z_i) = \frac{\exp(z_i)}{\sum_j e\,\mathrm{xp}\,(z_i)}$$

- використовується для останнього шару глибоких нейронних мереж для задач класифікації
- для навчання нейронної мережі при цьому в якості функції втрат використовується перехресна енропія.

Функція втрат Кульбака-Лейблера

Розбіжність визначається формулою: $D_{\mathrm{KL}}(\hat{p}_{\mathrm{data}} \| p_{\mathrm{model}}) = \mathbb{E}_{\mathbf{x} \sim \hat{p}_{\mathrm{data}}} [\log \hat{p}_{\mathrm{data}}(\boldsymbol{x}) - \log p_{\mathrm{model}}(\boldsymbol{x})]$

Перший член різниці залежить тільки від початкових даних, а не від моделі.

При навчанні моделі, мінімізуючи розбіжність КЛ, ми повинні мінімізувати величину:

$$D_{\mathrm{KL}}(\hat{p}_{\mathrm{data}} \| p_{\mathrm{model}}) = \mathbb{E}_{\mathbf{x} \sim \hat{p}_{\mathrm{data}}} [\log \hat{p}_{\mathrm{data}}(\boldsymbol{x}) - \log p_{\mathrm{model}}(\boldsymbol{x})]$$

$$- \mathbb{E}_{\mathbf{x} \sim \hat{p}_{\mathrm{data}}} [\log p_{\mathrm{model}}(\boldsymbol{x})]$$

Мінімізація розбіжності КЛ відповідає мінімізації перехресної ентропії між розподілами.

Стохастичний градієнтний спуск

1 Алгоритм стохастичного градієнта:

Вхід:

X^l — навчальна вибірка

η — темп навчання

λ параметр згладжування функціоналу Q

Вихід:

Вектор ваг w

Тіло:

2 Ініціалізувати ваги w_j $j = 0, .., n$.

Ініціалізувати поточну оцінку функціоналу:

$$Q := \sum_{i=1}^{l} L(a(x_i, w), y_i)$$

3 Повторювати:

Вибрати об'єкт x_i із X^l (наприклад, випадковим чином);

Обчислити вихідне значення алгоритму $a(\boldsymbol{x}_i, w)$ та помилку:

$$\varepsilon_i := L(a(x_i, w), y_i);$$

4 Зробити крок градієнтного спуску:

$$w := w - \eta L'_a(a(x_i, w), y_i)\varphi'(< w, x_i >)x_i$$

5 Оцінити значення функціоналу:

$$Q := (1 - \lambda)Q + \lambda\varepsilon_i$$

6 Поки значення Q не стабілізується та/або ваги w не припинять змінюватись.

Метод Адам

Adam – алгоритм оптимізації з адаптивною швидкістю навчання.

Алгоритм:

Require: величина кроку ε (за замовчуванням 0.001).

Require: коефіцієнти експоненціального затухання для оцінок моментів ρ1 і ρ2, що належать діапазону [0, 1) (за замовчуванням 0.9 і 0.999 відповідно).

Require: невелика константа δ для забезпечення чисельної стійкості.

Require: початкові значення параметрів θ.

Ініціалізувати змінні для першого і другого моментів s = 0, r = 0

Ініціалізувати крок за часом t = 0

while критерій зупинки не виконано do

Вибрати з навчального набору міні-пакет m прикладів {x (1), ..., x (m)} і мітки y (i).

Обчислити градієнт: g ← (1 / m) ∇θΣi L (f (x (i); θ), y (i)).

t ← t + 1

Оновити зміщену оцінку першого моменту: s ← ρ1s + (1 − ρ1) g Оновити зміщену оцінку другого моменту: r ← ρ2r + (1 − ρ2) g ⊙ g

Скорегувати зміщення першого моменту:

$$\hat{s} \leftarrow \frac{s}{1-\rho_1^t}$$

Скорегувати зміщення другого моменту:

$$\hat{r} \leftarrow \frac{r}{1-\rho_2^t}$$

Обчислити оновлення:

$$\Delta\theta = -\varepsilon \frac{\hat{s}}{\sqrt{\hat{r}} + \delta}$$

Застосувати оновлення: θ ← θ + Δθ. end while.

Метод Глоро (2013) і Хе (2015) для ініціалізації ваг

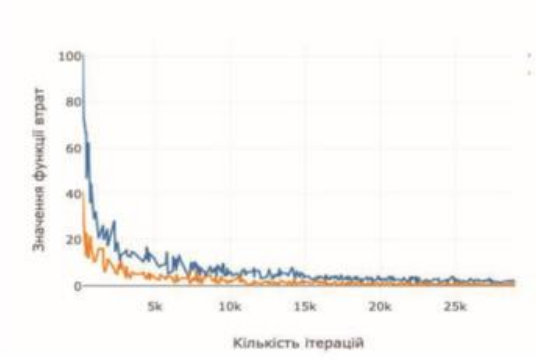
Розглянемо значення одного нейрону (до застосування функції активації):

$$y = w^{\top} x + b = \sum_i w_i x_i + b,$$

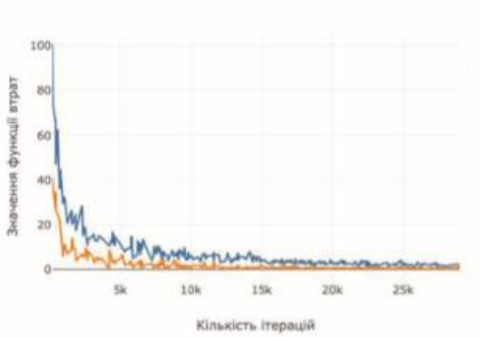
Де x – вектор вхідних значень, w – вектор параметрів. Таким чином, дисперсія *V ar(y)* не залежить від члена b, а залежить тільки від вектора вхідних значень і вектора параметрів. Позначимо і-й член суми як $y_i = w_i x_i$. Припустимо, що x_i та w_i – незалежні, маємо:

$$V\,ar(y_i) = V\,ar(w_i x_i) = E[w_i x_i] - (M[w_i x_i])^2$$
$$= M[x_i]^2\,V\,ar(w_i) + M[w_i]^2 V\,ar(x_i)$$
$$+ V\,ar(x_i) V\,ar(w_i)$$

Результат роботи методів Глоро і Хе



Порівняння навчання нейронної мережі на базі цифр MNIST з використанням ініціалізації Глоро



Порівняння навчання нейронної мережі на базі цифр MNIST з використанням ініціалізації Хе.

Приклад структури згорткової мережі CNN

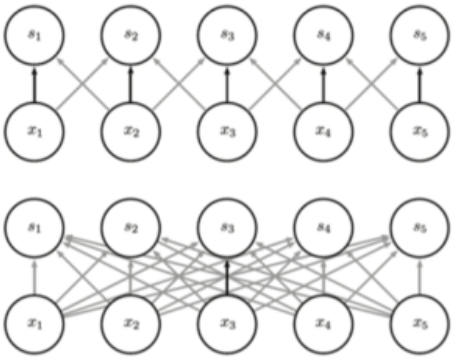
На вхід подається зображення, на виході маємо ймовірності належності цього зображення до конкретних класів

Розділення параметрів у згортковій нейронній мережі CNN

- В CNN один і той же параметр може використовуватися в декількох функціях однієї моделі.
- В CNN кожний елемент ядра застосовується до кожної позиції входу,

на відміну від традиційної нейронної мережі, де кожен елемент матриці використовується лише один раз, коли множиться на елемент вхідного сигналу при обчисленні вихідного сигнала шару.

- Згортка вважається значно ефективнішою у порівнянні з множенням матриць.



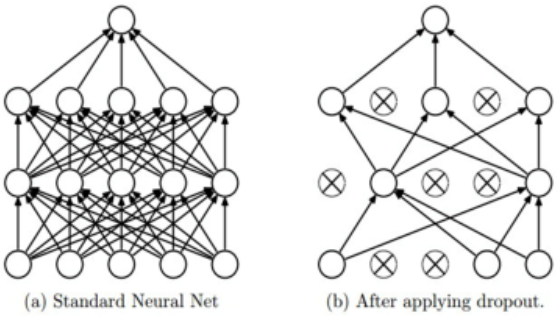
Чорними стрілками показано зв'язки, в яких приймає участь конкретний параметр в двох різних моделях.

Перенавчання (Overfitting)

Одна з проблем згорткових нейронних мереж (Convolution Neural Networks, CNN), складається в наступному: модель добре пояснює **тільки** приклади з навчальної вибірки, адаптуючись до навчальних прикладів, замість того щоб вчитися класифікувати приклади, які не брали участі в навчанні (втрачаючи здатність до узагальнення).

Найбільш якісне рішення цієї проблеми - Dropout

Метод Dropout для зменшення перенавчання моделі

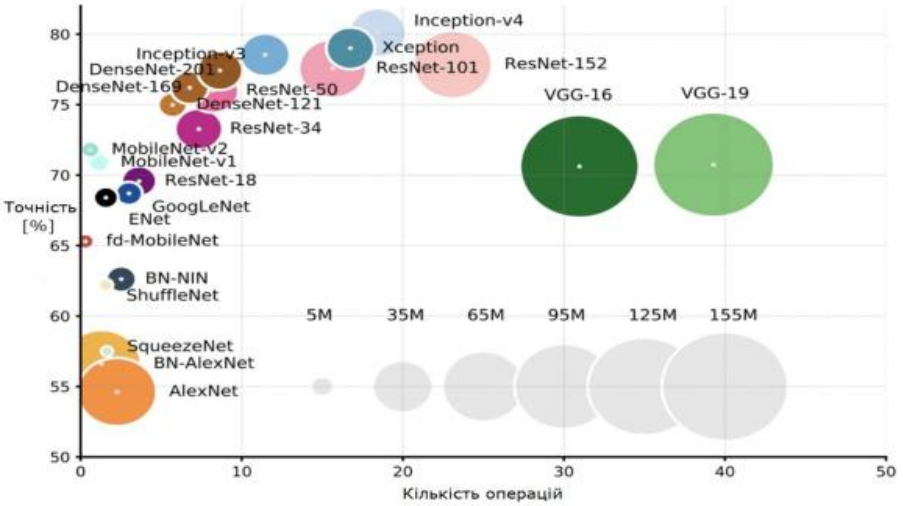


Графічне порівння мереж без та із застосуванням dropout

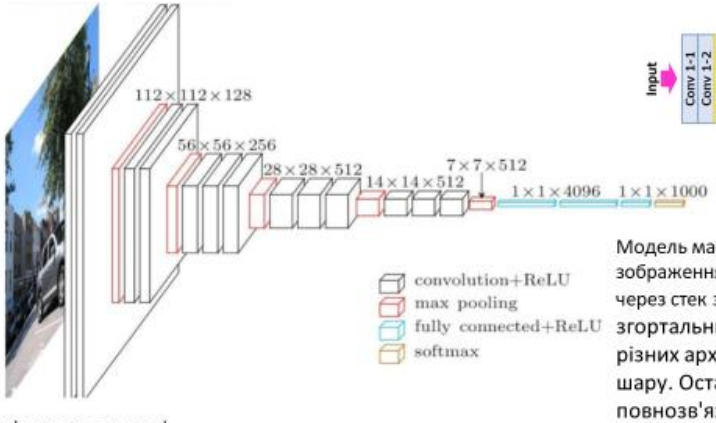
Головна ідея Dropout - замість навчання однієї нейронної мережи навчити ансамбль декількох , а потім усереднити отримані результати.

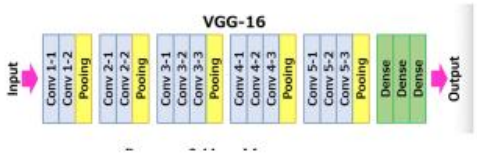
Мережі для навчання виходять за допомогою **виключення** з мережі (dropping out) нейронів з ймовірністю p виключені нейрони не вносять свій внесок в процес навчання ні на одному з етапів алгоритму зворотного поширення помилки (backpropagation); тому виключення хоча б одного з нейронів рівносильно навчання нової нейронної мережі.

Спеціалізовані згорткові мережі для отримання ознак зображень



Спеціалізована згорткова мережа VGG16



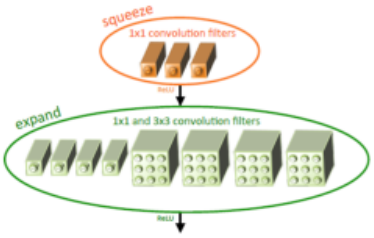


Архітектура моделі

Модель має на вході на шар conv1 подаються зображення RGB 224x224. Далі зображення проходять через стек згортальних шарів. Після стека згортальних шарів (який має різну глибину в різних архітектурах) йдуть три повнозв'язних шару. Останнім йде soft-max шар. Конфігурація повнозв'язних верств одна і та ж у всіх нейросетях.

Спеціалізована згорткова мережа SqueezeNet

CNN моделі має декілька різних стратегій оптимізація відносно інших моделей
Ключова з них, реалізована в даній моделі, це зменшення кількості вхідних каналів до 3 × 3 фільтрів , за допомогою шарів стискання (burn modul)



Модуль стискання з гіперпараметрами: s1 x 1 = 3, e1 x 1 =4, e3 x 3 =5

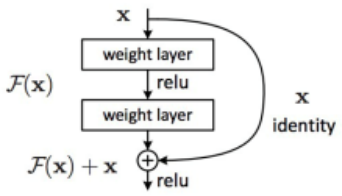
Burn modul зображенний на рисунку справа складається з шару згортки стискання (лише з фільтрами 1x1), який подається в шар розширення, який складається з міксу фільтрів згортки 3 x 3 та 1 x 1.

Архітектура мережі --->>>

Спеціалізована згорткова мережа ResNet

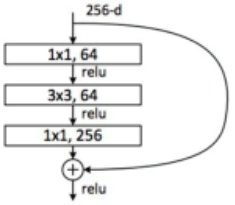
Робота архітектури ResNet виділяється наступним чином:

- на вихід подаються 2 згорткових шари
- обхід 2-х згорткових шарів



Існують різні варіації кількості шарів мережи. Кожен блок мережи має два рівня глибини (для невеликих мереж як ResNet 18,34) або 3 рівня (ResNet 50, 101, 152).

На далі, в експериментах застовується мережа із 50 шарами
ResNet50 - кожен 3-шаровий блок замінюється в 34-шарової мережі цим 3-шаровим вузьким місцем, в результаті виходить 50-шарова ResNet (див. Шар ~вузького місця~). Ця модель має 3,8 мільярда FLOPs.



←Шар ~вузького місця~

Використані метрики для вибору найкращої моделі класифікації

1.Accuracy

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

2.F1-міра

$$F = \frac{2 * precision * recall}{precision + recall}$$

3.Precision

$$precision = \frac{TP}{TP + FP}$$

4.Recall

$$recall = \frac{TP}{TP + FN}$$

**TP - істинно позитивно*
**TN - істинно негативно*
**FP - помилково позитивно*
**FN - помилково негативно*

Використані бібліотеки середовища Python

Tensorflow – open-source бібліотека для машинного навчання та роботи с нейронними мережами

Sklearn – також бібліотека для машинного-навчання, у нашему випадку використовується для виведення даних якості навчання та препроцесінгу

Matplotlib.pyplot – ілюстрація графіків

Pandas – бібліотека для роботи, аналізу та маніпуляцій з набором даних

NumPy – научно-технічні обчислювання

Обробка зображень та розділ даних на тренувальні та тестові

```
print("[INFO] loading images...")
imagePaths = list(paths.list_images(dataset_path))
data = []
labels = []

for imagePath in imagePaths:

    label = imagePath.split(os.path.sep)[-2]

    image = cv2.imread(imagePath)
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    image = cv2.resize(image, (224, 224))

    data.append(image)
    labels.append(label)

data = np.array(data) / 255.0
labels = np.array(labels)

lb = LabelBinarizer()
labels = lb.fit_transform(labels)
labels = to_categorical(labels)

(trainX, testX, trainY, testY) = train_test_split(data, labels, test_size=0.20, stratify=labels, random_state=42)

trainAug = ImageDataGenerator(rotation_range=15, fill_mode="nearest")
```

Релізація моделі Squeezenet

Модуль стискання

```
def fire(x, squeeze, expand):
  y  = tf.keras.layers.Conv2D(filters=squeeze, kernel_size=1, activation='relu', padding='same')(x)
  y = tf.keras.layers.BatchNormalization(momentum=bnmomemtum)(y)
  y1 = tf.keras.layers.Conv2D(filters=expand//2, kernel_size=1, activation='relu', padding='same')(y)
  y1 = tf.keras.layers.BatchNormalization(momentum=bnmomemtum)(y1)
  y3 = tf.keras.layers.Conv2D(filters=expand//2, kernel_size=3, activation='relu', padding='same')(y)
  y3 = tf.keras.layers.BatchNormalization(momentum=bnmomemtum)(y3)
  return tf.keras.layers.concatenate([y1, y3])

def fire_module(squeeze, expand):
  return lambda x: fire(x, squeeze, expand)
```

Реалізація архітектури моделі----->>>>>

```
x = tf.keras.layers.Input(shape=[224, 224, 3])

y = tf.keras.layers.Conv2D(kernel_size=3, filters=32, padding='same', use_bias=True, activation='relu')(x)
y = tf.keras.layers.BatchNormalization(momentum=bnmomemtum)(y)
y = fire_module(24, 48)(y)
y = tf.keras.layers.MaxPooling2D(pool_size=2)(y)
y = fire_module(48, 96)(y)
y = tf.keras.layers.MaxPooling2D(pool_size=2)(y)
y = fire_module(64, 128)(y)
y = tf.keras.layers.MaxPooling2D(pool_size=2)(y)
y = fire_module(48, 96)(y)
y = tf.keras.layers.MaxPooling2D(pool_size=2)(y)
y = fire_module(24, 48)(y)
y = tf.keras.layers.GlobalAveragePooling2D()(y)
y = tf.keras.layers.Dropout(0.5)(y)
y = tf.keras.layers.Dense(2, activation='softmax')(y)

model = tf.keras.Model(x, y)
```

Побудова моделі класифікації зображень рентгенів здорової людини і хворої на COVID19

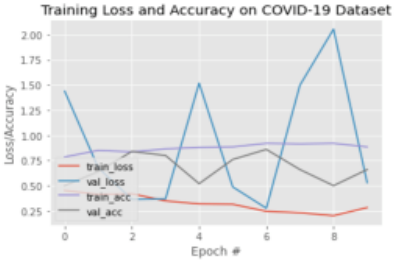
Squezenet результати із використанням loss = ‘binary_crossentropy’

На тестовій вибірці

	precision	recall	f1-score
covid	0.60	0.96	0.74
normal	0.90	0.36	0.51
accuracy			0.66
macro avg	0.75	0.66	0.63
weighted avg	0.75	0.66	0.63

Матриця помилок

```
[[24  1]
 [16  9]]
acc: 0.6600
sensitivity: 0.9600
specificity: 0.3600
```



Результати навчання за епохами

Squezenet + 'binary_cross'	loss	acc	val_loss	val_acc
EPOCH 1	0.4522	0.785	1.4373	0.5
EPOCH 2	0.4135	0.85	0.6771	0.64
EPOCH 3	0.4203	0.835	0.3639	0.84
EPOCH 4	0.3488	0.865	0.3694	0.8
EPOCH 5	0.3192	0.88	1.517	0.52
EPOCH 6	0.3158	0.885	0.4879	0.76
EPOCH 7	0.2454	0.92	0.2759	0.86
EPOCH 8	0.2312	0.915	1.4971	0.66
EPOCH 9	0.2024	0.92	2.0532	0.5
EPOCH 10	0.2826	0.885	0.5303	0.66

Побудова моделі класифікації зображень рентгенів здорової людини і хворої на COVID19

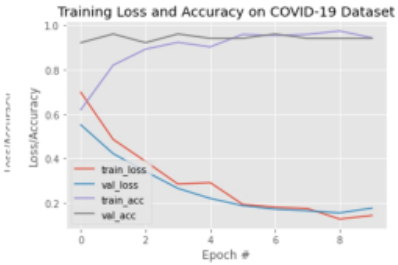
VGG16 результати із використанням loss = ‘binary_crossentropy’

На тестовій вибірці

	precision	recall	f1-score
covid	0.89	1.00	0.94
normal	1.00	0.88	0.94
accuracy			0.94
macro avg	0.95	0.94	0.94
weighted avg	0.95	0.94	0.94

Матриця помилок

```
[[25  0]
 [ 3 23]]
acc: 0.9412
sensitivity: 1.0000
specificity: 0.8846
```



Результати навчання за епохами

	loss	acc	val_loss	val_acc
EPOCH 1	0.698221	0.620000	0.552068	0.921569
EPOCH 2	0.486643	0.820513	0.422721	0.960784
EPOCH 3	0.387653	0.892308	0.342231	0.921569
EPOCH 4	0.285871	0.923077	0.266439	0.960784
EPOCH 5	0.292237	0.902564	0.221258	0.941176
EPOCH 6	0.194723	0.958974	0.188820	0.941176
EPOCH 7	0.181866	0.953846	0.172962	0.960784
EPOCH 8	0.176088	0.958974	0.165455	0.941176
EPOCH 9	0.128608	0.974359	0.156087	0.941176
EPOCH 10	0.144520	0.943590	0.177703	0.941176

```
opt = Adam(lr=INIT_LR, decay=INIT_LR / EPOCHS)
```

```
model.compile(loss="binary_crossentropy", optimizer=opt, metrics=["accuracy"])
```

Побудова моделі класифікації зображень рентгенів здорової людини і хворої на COVID19

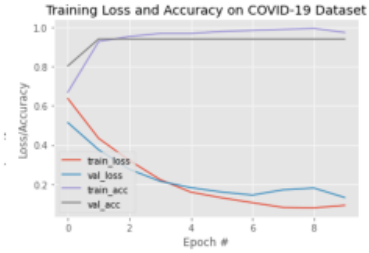
VGG16 результати без використання дропауту

На тестовій вибірці

	precision	recall	f1-score
covid	0.92	0.96	0.94
normal	0.96	0.92	0.94
accuracy			0.94
macro avg	0.94	0.94	0.94
weighted avg	0.94	0.94	0.94

Матриця помилок

```
[[24  1]
 [ 2 24]]
acc: 0.9412
sensitivity: 0.9600
specificity: 0.9231
```



Результати навчання за епохами

	loss	acc	val_loss	val_acc
EPOCH 1	0.636441	0.670000	0.513700	0.803922
EPOCH 2	0.433476	0.928205	0.375375	0.941176
EPOCH 3	0.320856	0.953846	0.275544	0.941176
EPOCH 4	0.223211	0.969231	0.214710	0.941176
EPOCH 5	0.159231	0.969231	0.182106	0.941176
EPOCH 6	0.130305	0.979487	0.159862	0.941176
EPOCH 7	0.105350	0.984615	0.144310	0.941176
EPOCH 8	0.081026	0.989744	0.171275	0.941176
EPOCH 9	0.078948	0.994872	0.180171	0.941176
EPOCH 10	0.091740	0.974359	0.131913	0.941176

Побудова моделі класифікації зображень рентгенів здорової людини і хворої на COVID19

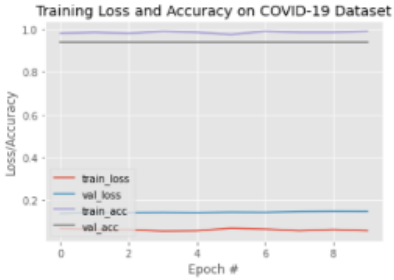
VGG16 результати із використанням opt = SGD (стохастичний градієт)

На тестовій вибірці

	precision	recall	f1-score
covid	0.89	1.00	0.94
normal	1.00	0.88	0.94
accuracy			0.94
macro avg	0.95	0.94	0.94
weighted avg	0.95	0.94	0.94

Матриця помилок

```
[[25  0]
 [ 3 23]]
acc: 0.9412
sensitivity: 1.0000
specificity: 0.8846
```



Результати навчання за епохами

	loss	acc	val_loss	val_acc
EPOCH 1	0.067897	0.980000	0.137923	0.941176
EPOCH 2	0.062733	0.984615	0.143203	0.941176
EPOCH 3	0.062508	0.979487	0.142211	0.941176
EPOCH 4	0.056181	0.989744	0.142906	0.941176
EPOCH 5	0.057666	0.984615	0.142089	0.941176
EPOCH 6	0.068904	0.974359	0.144000	0.941176
EPOCH 7	0.064900	0.989744	0.143259	0.941176
EPOCH 8	0.057887	0.984615	0.147036	0.941176
EPOCH 9	0.062723	0.984615	0.148193	0.941176
EPOCH 10	0.058083	0.989744	0.147744	0.941176

```
opt = SGD(
    learning_rate=INIT_LR, momentum=0.0, nesterov=False, name='SGD'
)
```

Побудова моделі класифікації зображень рентгенів здорової людини і хворої на COVID19

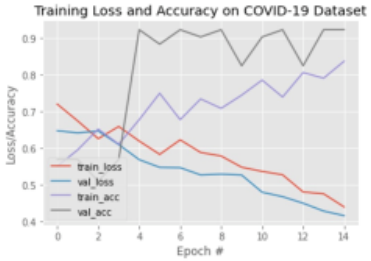
ResNet50 результати із використанням loss = ‘binary_crossentropy’

На тестовій вибірці

	precision	recall	f1-score
covid	0.96	0.88	0.92
normal	0.89	0.96	0.93
accuracy			0.92
macro avg	0.92	0.92	0.92
weighted avg	0.92	0.92	0.92

Матриця помилок

```
[[22  3]
 [ 1 25]]
acc: 0.9216
sensitivity: 0.8800
specificity: 0.9615
```



Результати навчання за епохами

	loss	acc	val_loss	val_acc
EPOCH 1	0.719181	0.550000	0.646954	0.568627
EPOCH 2	0.673448	0.594872	0.640986	0.568627
EPOCH 3	0.625215	0.651282	0.646157	0.509804
EPOCH 4	0.658369	0.610256	0.609318	0.568627
EPOCH 5	0.618812	0.676923	0.568136	0.921569
EPOCH 6	0.582537	0.748718	0.547258	0.882353
EPOCH 7	0.622050	0.676923	0.546408	0.921569
EPOCH 8	0.587702	0.733333	0.526610	0.901961
EPOCH 9	0.578113	0.707692	0.528932	0.921569
EPOCH 10	0.547635	0.743590	0.526445	0.823529
EPOCH 11	0.536269	0.784615	0.479625	0.901961
EPOCH 12	0.527021	0.738462	0.467544	0.921569
EPOCH 13	0.480324	0.805128	0.449752	0.823529
EPOCH 14	0.475362	0.789744	0.428393	0.921569
EPOCH 15	0.439648	0.835897	0.416012	0.921569

Побудова моделі класифікації зображень рентгенів здорової людини і хворої на COVID19

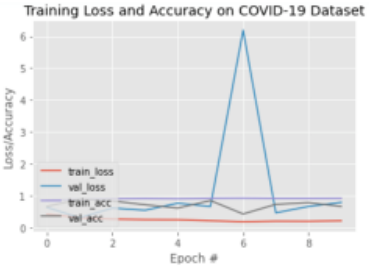
ResNet50 результати без використання дропауту

На тестовій вибірці

	precision	recall	f1-score
covid	0.86	0.61	0.71
normal	0.50	0.80	0.62
accuracy			0.67
macro avg	0.68	0.70	0.66
weighted avg	0.74	0.67	0.68

Матриця помилок

```
[[24  1]
 [16  9]]
acc: 0.6600
sensitivity: 0.9600
specificity: 0.3600
```

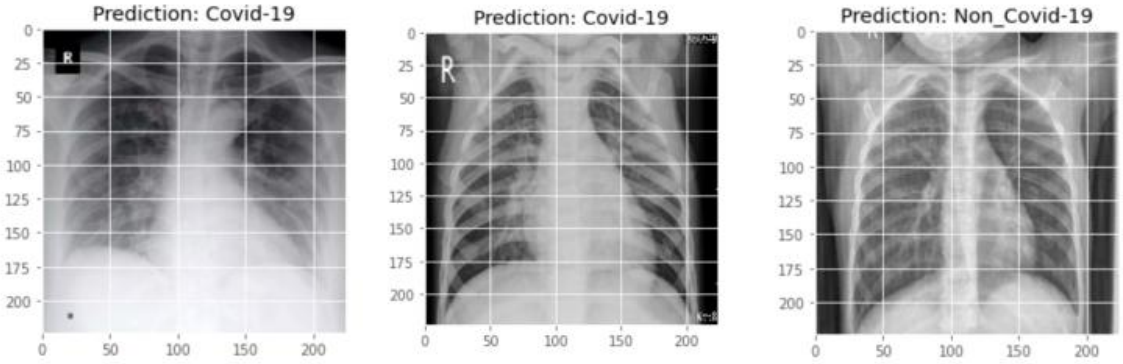


Результати навчання за епохами

	loss	acc	val_loss	val_acc
EPOCH 1	0.432569	0.861486	0.643321	0.671053
EPOCH 2	0.314589	0.887372	0.313614	0.894737
EPOCH 3	0.271643	0.907850	0.619817	0.842105
EPOCH 4	0.259011	0.921502	0.551127	0.723684
EPOCH 5	0.255940	0.907850	0.772096	0.618421
EPOCH 6	0.223079	0.928328	0.668588	0.855263
EPOCH 7	0.191730	0.931741	6.191057	0.434211
EPOCH 8	0.209370	0.924915	0.471527	0.736842
EPOCH 9	0.206348	0.924915	0.669967	0.789474
EPOCH 10	0.221920	0.928328	0.797436	0.671053

На основі обраної моделі з найкращими показниками - спеціалізованої нейронної мережі VGG16

	precision	recall	f1-score
covid	0.89	1.00	0.94
normal	1.00	0.88	0.94
accuracy			0.94



Виявлення факторів, які впливають на перебіг захворювання COVID19

Класифікація відбувається на знімках хворих , які знаходяться у сладному стані, отже класифікування зображень, як хворого, ще й з високою ймовірністю є фактором тяжкого перебігу захворювання

Результати експериментів

1. Досліджено **спеціалізовані моделі згорткових нейронних мереж** , які відрізняються:

• різними параметрами шарів

• включення та виключення Dropout

• використанням різних функцій втрат

2. Виконано прогноз діагнозу на основі найкращої моделі VGG-16, RESNET50 та Squeezenet із використанням бінарної кросентропії.

Висновки по роботі

- Проведено дослідження та проаналізувати ситуацію з поширенням коронавірусу в різних країнах світу
- Проведено аналіз методів класифікації зображень
- Побудовано моделі класифікації зображень рентгенів здорової людини і хворої на COVID19
- Виконано прогнозування діагнозу COVID19 на основі вибраної найкращої моделі
- Виявлено фактори, які впливають на перебіг захворювання
- Розроблено програмний продукт для прогнозування діагнозу COVID19 на основі вибраної найкращої моделі

Перспективи подальших досліджень

1. Реалізація мобільного додатку або веб-сайту, з можливістю завантаження свого зображення, та отримання прогнозу наявності захворювання

2. Класифікування типу коронавірусу або його тяжкості.

Дипломна робота

Моделі та методи прогнозування поширення COVID-19. Прогнозування поширення коронавірусу та аналіз впливу карантинних мір в різних країнах світу

Виконала:
Лупаненко Софія Олександрівна
Науковий керівний:
к.т.н. доцент Недашківська Надія Іванівна

Об’єкт, предмет, ціль дослідження

Об’єкт дослідження: кількість хворих, представлені в часовому ряді.

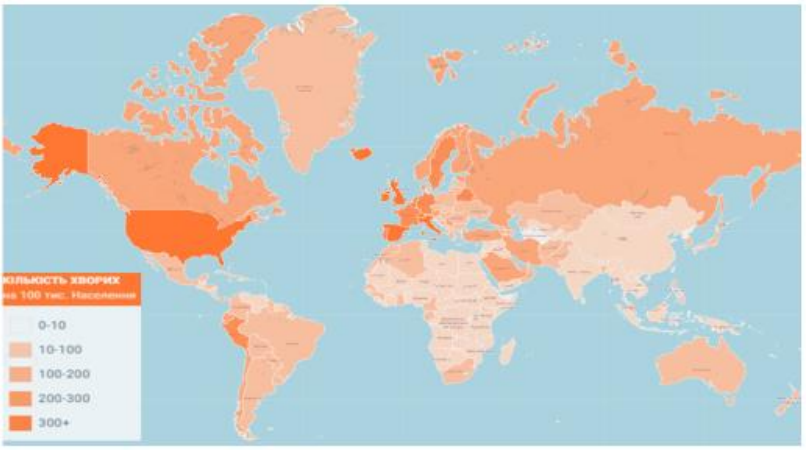
Предмет дослідження: моделі аналізу часових рядів на основі методів регресії, опорних векторів, нейронних мереж та ансамблів моделей.

Ціль дослідження: порівняльний аналіз методів аналізу часових рядів, побудова прогнозу поширення коронавірусу на основі вибраної найкращої моделі.

Актуальність

- Контролювання поширення вірусу в країнах Світу;
- Дослідження кращої моделі для прогнозування;
- Дослідження впливу карантина на поширення.

Карта кількості випадків коронавірсу у Світі



Вік	Кількість померлих	% смертей	З основними симптомами	Без основних симптомів	Симптоми невідомі	Частка загиблих невідомих
0 – 17	3	0,04%	3	0	0	**0%**
18 – 44	309	4,5%	244	25	40	**1,0%**
45 – 64	1581	23,1%	1343	59	179	**3,5%**
65 – 74	1683	24,6%	1,272	26	385	**6,0%**
75+	3263	47,7%	2289	27	947	**14,2%**

Стать	Смертей	%	Симптоми	%	Без симптом	%	Симптоми невідомі	%
Ч	4095	61,8%	3087	62,2%	96	72,2%	912	59,5%
Ж	2530	38,2%	1,887	37,8%	37	27,8%	620	40,5%

Постановка задачі

1. **Порівняльний аналіз методів** аналізу часових рядів:
- моделей поліноміальної регресії,
- методу опорних векторів,
- багатошарових нейронних мереж прямого розповсюдження,
- ансамблей моделей.

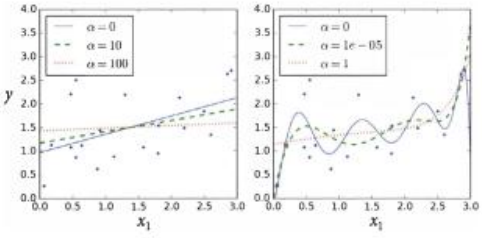
2. **Побудова і дослідження моделей** часових рядів для прогнозування поширення коронавірусу в різних країнах світу.

3. **Побудова прогнозу** поширення коронавірусу на основі вибраної найкращої моделі.

4. **Написання програмного продукту** для побудови, дослідження моделей і знаходження прогнозу.

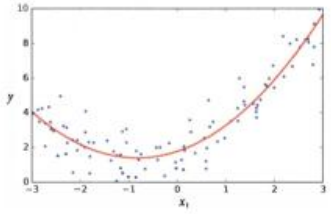
Регуляризовані моделі лінійної регресії

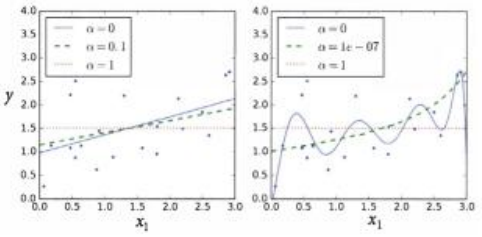
- Гребенева регресія $\alpha \sum_{i=1}^{n} \theta_i^2$



- Лассо-регресія $J(\theta) = MSE(\theta) + \alpha \sum_{i=1}^{n} |\theta_i|$

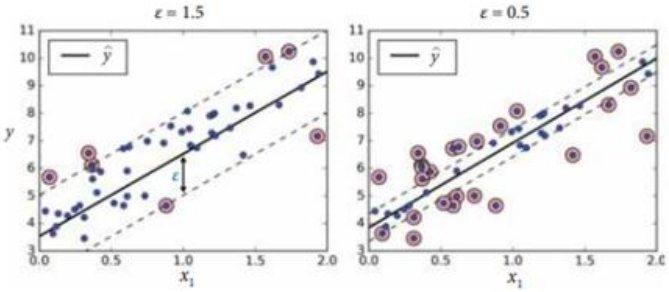
- Поліноміальна регресія





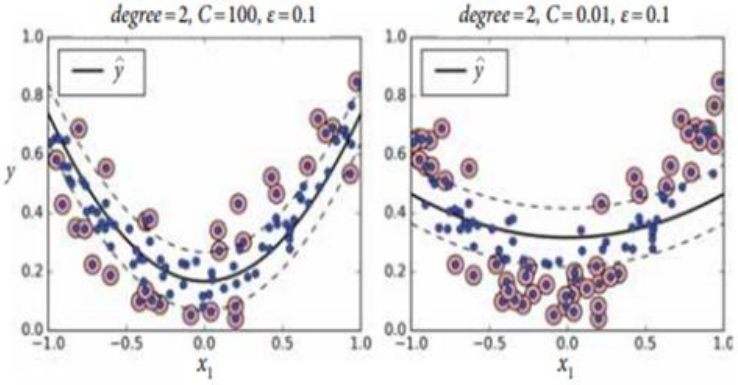
Метод опорних векторів (SVM)

- Існує більш ніж одна площина, яка розділяє два класи.
- Замість того, щоб розділяти класи площиною, будемо малювати біля кожної площини відступ (margin).



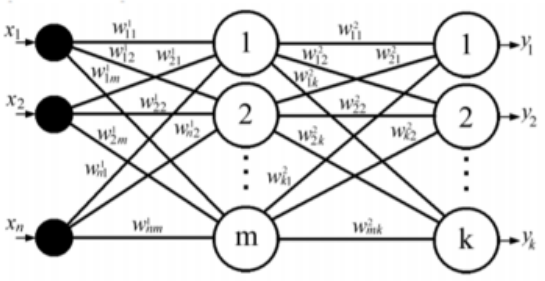
- Відступ має деяку ширину до найближчої навчальної точки.
- В якості оптимальної моделі вибирається площина, яка максимізує цей відступ, тобто ширину смуги.

Регресія SVM, яка застосовує поліноміальне ядро 2-го порядку.



Побудова багатошарової нейронної мережі

1. Вибрати початкову конфігурацію мережі.
2. Провести ряд експериментів з різними конфігураціями.
3. Якщо в черговому експерименті спостерігається неповне навчання, спробувати додати додаткові нейрони або проміжний шар.
4. Якщо маємо в результаті перенавчання, то видаляємо приховані елементи.



Нейронна мережа для регресії

Багатошаровий персептрон. Клас MLPRegression.

Приклад реалізації:

```
model = MLPRegressor(hidden_layer_sizes=[32, 26, 10, ],
max_iter=100000, alpha=0.0005, random_state=26,  solver='lbfgs',
learning_rate='constant', validation_fraction=0.1)
```

Методи регуляризації параметрів нейронної мережі

L1-регуляризація параметрів моделі w визначається за формулою: $\Omega(\theta) = \|w\|_1 = \sum_i |w_i|$

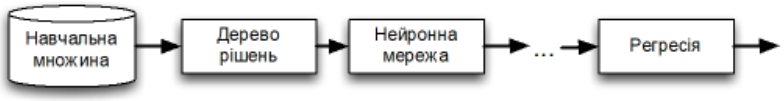
L1-регуляризація параметрів моделі w визначається за формулою: $\Omega(\theta) = \alpha \sum_{i=1}^{n} \theta_i^2$

Метод зменшення ваг: $\Omega(\theta) = \frac{1}{2}\|w\|$

Ансамбль моделей

Ансамблеве навчання – це метод машинного навчання, де декілька моделей поєднуються для отримання кращих результа

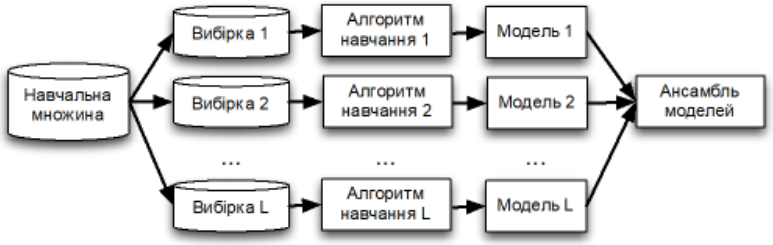
Однорідний ансамбль



Ансамбль з моделей різного типу

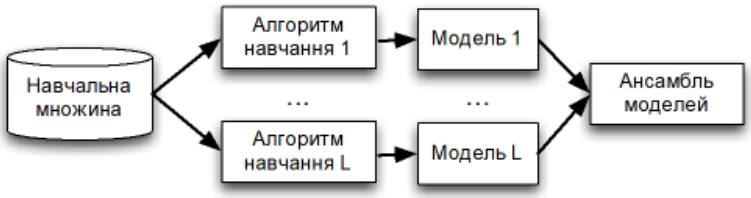
Перевага – додаткова гнучкість. **Недолік** – необхідні додаткові перетворення для узгодження входів і виходів моделей різного типу.

Види ансамблів моделей



Перевибірка (resampling)

Використання єдиної навчальної множини для всіх моделей ансамбля



Беггінг (покращуюче об'єднання, bootstrap aggregating)

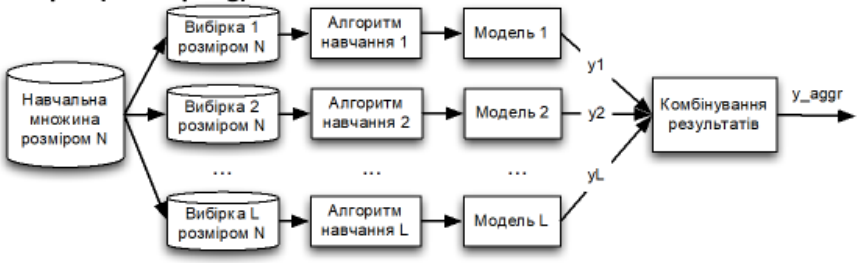
В основі беггінгу – технологія "збурення та комбінування"

- Формування декількох вибірок на основі навчальної множини

- Додавання шуму
- Адаптивне зважування
- Випадковий вибір між конкуруючими вузлами (розбиттями)

Беггінг (продовження)

Перевибірка (resampling)



Етапи:

- Формування вибірок однакового розміру випадковим чином на основі навчальної множини
- Побудова моделі на основі кожної вибірки
- Комбінування результатів

Бустінг (підвищення, boosting)

- Моделі створюються послідовно. Починає створення ансамбля на основі єдиної навчальної множини.
- Кожна нова модель ансамбля будується на основі результатів раніше побудованих моделей.
- Нові моделі будуються таким чином, щоб вони доповнювали раніше побудовані моделі, виконували ту роботу, яку інші моделі не змогли зробити на попередніх кроках.
- Кожній моделі ансамбля залежно від її точності присвоюється вага.

Недоліки бустінгу:

- Приклади з низькими вагами не потрапляють на наступну ітерацію. Наслідок втрата частини корисної інформації.
- Більша схильність до перенавчання в порівнянні з беггінгом. Кількість ітерацій L має бути якомога меншою без втрати точності.
- Низька прозорість ансамблю моделей для аналітика (недолік властивий і беггінгу).
- Складність інтерпретації результатів (недолік властивий і беггінгу).

Стекінг

Його ідея полягає в тому, щоб навчити кілька різних слабких моделей та поєднати їх, навчаючи мета-модель для виведення прогнозів на основі множинних прогнозів цих моделей.

Ансамбль стекінгу складається з L моделей.

Хід роботи:

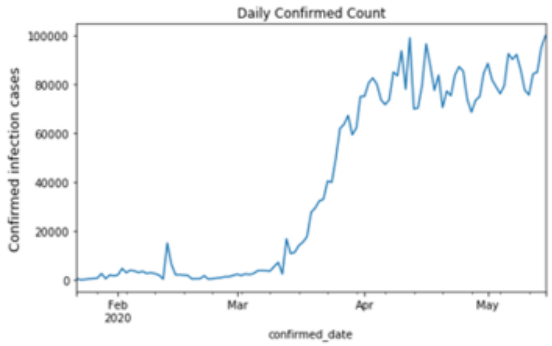
- розділити дані тренувань у два рази;
- вибирайте L слабких учнів і пристосовуйте їх до даних першого згину;
- для кожного з моделей, що склалися, зробіть прогнози для спостережень у другій частині;
- розмістити мета-модель на другій складці, використовуючи прогнози, зроблені слабкими моделями як вхідні дані.

Вхідні часові ряди

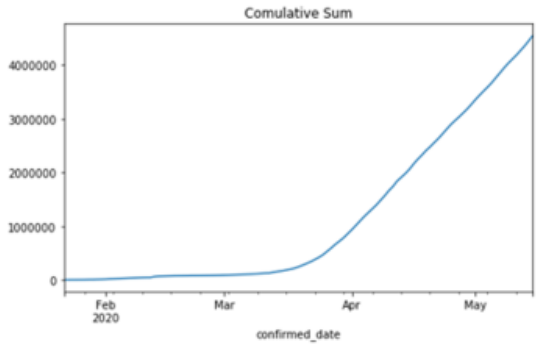
Id,Province_State,Country_Region,Date,Confirmed_date,Fatalities
1,,Afghanistan,2020-01-22,0.0,0.0
2,,Afghanistan,2020-01-23,0.0,0.0
3,,Afghanistan,2020-01-24,0.0,0.0
4,,Afghanistan,2020-01-25,0.0,0.0
5,,Afghanistan,2020-01-26,0.0,0.0
6,,Afghanistan,2020-01-27,0.0,0.0
7,,Afghanistan,2020-01-28,0.0,0.0
8,,Afghanistan,2020-01-29,0.0,0.0
9,,Afghanistan,2020-01-30,0.0,0.0
10,,Afghanistan,2020-01-31,0.0,0.0

Вхідні часові ряди

Кількість нових випадків заражень **у світі** щоденно

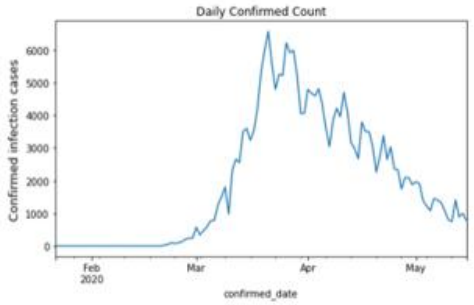


Сумарна кількість випадків заражень **у світі** за весь час

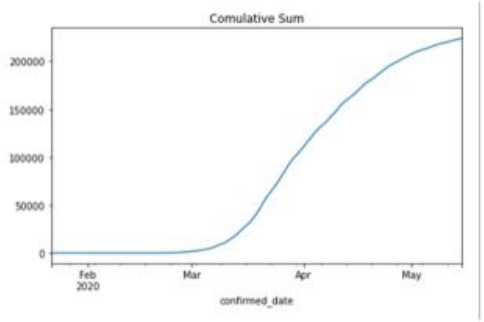


Вхідні часові ряди

Кількість нових випадків заражень щоденно



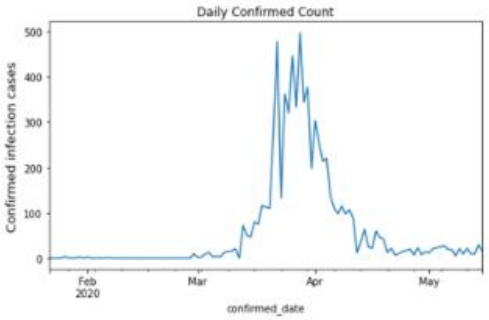
Сумарна кількість випадків заражень за весь час



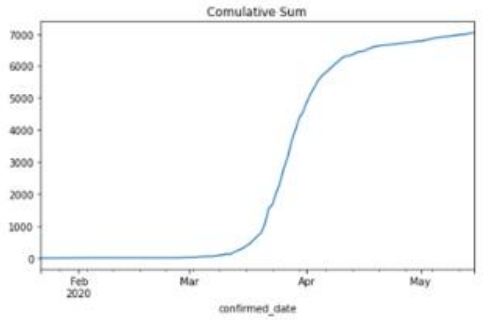
Італія

Вхідні часові ряди

Кількість нових випадків заражень щоденно



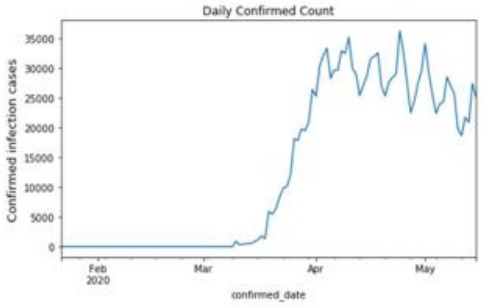
Сумарна кількість випадків заражень за весь час



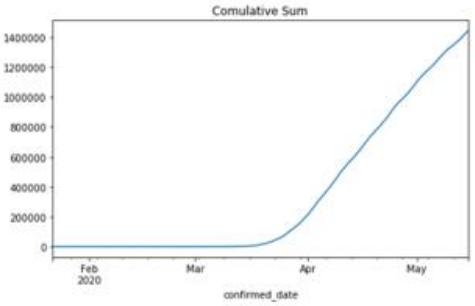
Австралія

Вхідні часові ряди

Кількість нових випадків заражень щоденно



Сумарна кількість випадків заражень за весь час



США

Навчання моделі нейронної мережі алгоритмом MLPRegressor

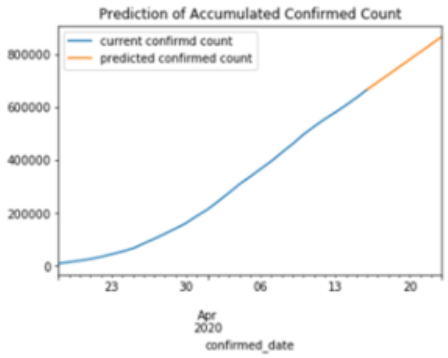
```
model = MLPRegressor(hidden_layer_sizes=[32, 26, 10, ], max_iter=100000, alpha=0.0005,
random_state=26,  solver='lbfgs',  learning_rate='constant',  validation_fraction=0.1)
```

Досліджувані параметри алгоритму:

- hidden_layer_sizes – розмір скритих шарів
- solver - метод розрахунку ваг: ‘lbfgs’ та ‘adam’
- f - функція активації для скритого шару
- alpha - параметр регуляризації
- learning_rate – швидкість навчання
- max_iter – максимальна кількість ітерацій

Прогноз кількості приросту захворювань на основі багатошарового персептрона

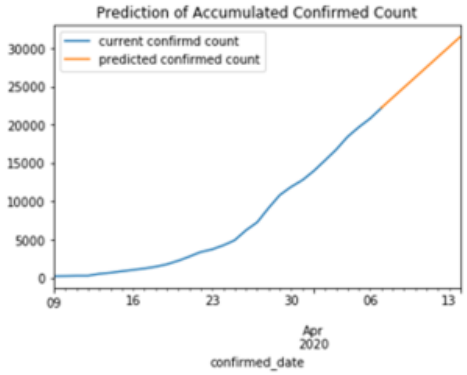
Прогноз кількості приросту захворювань для США




```
Predicted data:
2020-04-17    695275
2020-04-18    723845
2020-04-19    752415
2020-04-20    780985
2020-04-21    809555
2020-04-22    838125
2020-04-23    866695
dtype: int32
Real data:
confirmed_date
2020-04-17    699541.0
2020-04-18    732031.0
2020-04-19    758920.0
2020-04-20    784160.0
2020-04-21    811699.0
2020-04-22    840054.0
2020-04-23    869004.0
```

R^2=0.99964

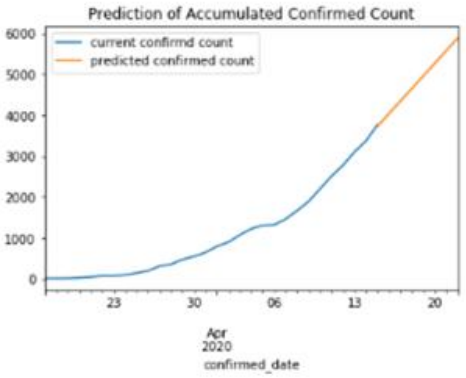
Прогноз кількості приросту захворювань для Бельгії




```
Predicted data:
2020-04-08    23543
2020-04-09    24868
2020-04-10    26193
2020-04-11    27518
2020-04-12    28843
2020-04-13    30168
2020-04-14    31493
dtype: int32
Real data:
confirmed_date
2020-04-08    23403.0
2020-04-09    24983.0
2020-04-10    26667.0
2020-04-11    28018.0
2020-04-12    29647.0
2020-04-13    30589.0
2020-04-14    31119.0
```

R^2=0.97917

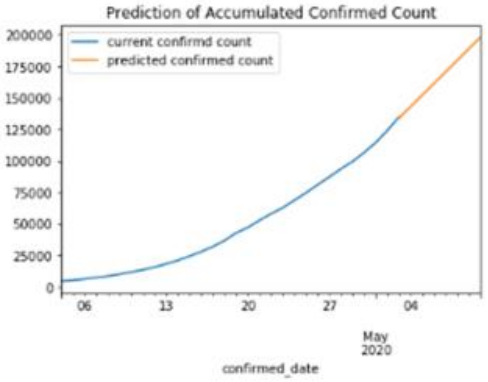
Прогноз для України




```
Predicted data:
2020-04-16    4036
2020-04-17    4344
2020-04-18    4652
2020-04-19    4960
2020-04-20    5268
2020-04-21    5576
2020-04-22    5884
dtype: int32
Real data:
confirmed_date
2020-04-16    4161.0
2020-04-17    4662.0
2020-04-18    5106.0
2020-04-19    5449.0
2020-04-20    5710.0
2020-04-21    6125.0
2020-04-22    6592.0
```

R^2=0.94074

Прогноз для Росії

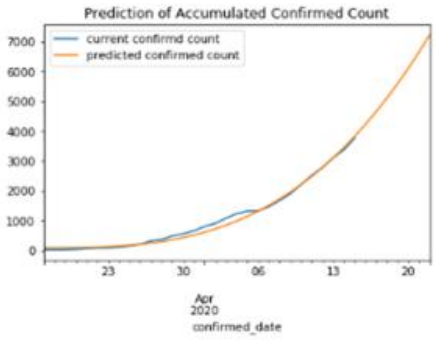



```
Predicted data:
2020-05-04    142931
2020-05-05    152098
2020-05-06    161265
2020-05-07    170432
2020-05-08    179599
2020-05-09    188766
2020-05-10    197933
dtype: int32
Real data:
confirmed_date
2020-05-04    145268.0
2020-05-05    155370.0
2020-05-06    165929.0
2020-05-07    177160.0
2020-05-08    187859.0
2020-05-09    198676.0
2020-05-10    209688.0
```

R^2=0.99964

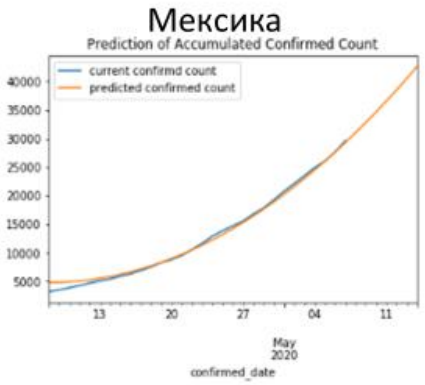
Прогноз кількості приросту захворювань на основі методу опорних векторів

Бельгія



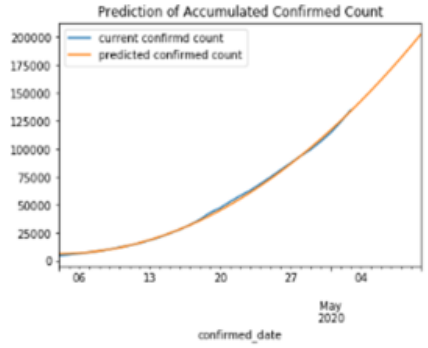
0.8769564806439968

Мексика



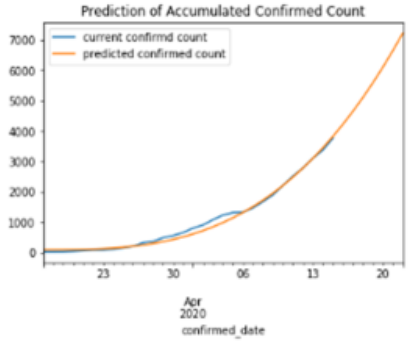
0.9902607244669575

Росія



0.9259077408231322

Україна



0.8769564806439968

Прогноз кількості приросту захворювань для на основі ансамбля моделей

Побудова ансамблів моделей

```
### Random Forest Model
rand_forest_model = RandomForestRegressor(n_estimators=1000, max_features=1, oob_score=True, random_state=115)
rand_forest_model.fit(x,y)
print('R-square value of Random Forest Model: ', rand_forest_model.score(test, y2))

### Gradient Boosting
gr_boosting_model = GradientBoostingRegressor(n_estimators=500, learning_rate=0.1, subsample=1, max_features=1, loss='ls')
gr_boosting_model.fit(x,y)
print('Gradient Boosting:', gr_boosting_model.score(test, y2))

# Ada Boosting Regressor
ad_boost_model = AdaBoostRegressor(base_estimator=None, n_estimators=50, learning_rate=1.0, loss='linear',
                                   random_state=None)
ad_boost_model.fit(x,y)
print('Ada Boosting Regressor:', ad_boost_model.score(test, y2))

# Bagging Regressor
bagging_model = BaggingRegressor(base_estimator=None, n_estimators=10, max_samples=1.0, max_features=1.0,
                                 bootstrap=True, bootstrap_features=False, oob_score=False, warm_start=False, n_jobs=None,
bagging_model.fit(x,y)
print('Bagging Regressor', bagging_model.score(test, y2))

# Voting Regressor
voting_model = VotingRegressor(estimators=100, weights=None, n_jobs=None, verbose=False)
voting_model.fit(x,y)
print('Voting Regressor', voting_model.score(test, y2))

# Stacking Regressor
stacking_model = StackingRegressor(estimators=100, final_estimator=None, cv=None, n_jobs=None,
                                   passthrough=False, verbose=0)
stacking_model.fit(x,y)
print('Stacking Regressor', stacking_model.score(test, y2))
```

Оцінки прогнозу на основі ансамблю моделей

```
R-square value of Random Forest Model: 0.9713020163118805
R-square value of Gradient Boosting: 0.984212591296279
R-square value of Ada Boosting Regressor: 0.9673130769242105
R-square value of Bagging Regressor: 0.9056376408076269
R-square value of Voting Regressor: 0.7239865312461513
R-square value of Stacking Regressor: 0.88398253176641513
```

Висновки

1. Виконано порівняльний аналіз моделей поліноміальної регресії, опорних векторів, багатошарових нейронних мереж прямого розповсюдження, ансамблів моделей.

2. Побудовано і досліджено моделі часових рядів для прогнозування поширення коронавірусу в різних країнах світу.

3. Побудовано прогнози поширення коронавірусу на основі вибраної найкращої моделі.

4. Реалізовано програмний продукт в середовищі python з використанням сучасних бібліотек Scikit-Learn та Pandas.

Дякую за увагу !